# **INTERNSHIP REPORT**

A report submitted in partial fulfilment of the requirements for the Award of Degree of

# BACHELOR OF ENGINEERING in COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

**Submitted by** 

SIVAA GANESH S

Reg. No.: 211011043

B.E. CSE (DS)

V Semester



# DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING FACULTY OF ENGINEERING AND TECHNOLOGY ANNAMALAI UNIVERSITY ANNAMALAI NAGAR - 608002

**NOVEMBER - 2023** 

# FACULTY OF ENGINEERING AND TECHNOLOGY DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# **CERTIFICATE**

This is to certify that the "Internship Report" submitted by SIVAA GANESH S Reg.No.: 211011043
is the work done by him/ her and submitted during the academic year 2023 – 2024, in partial fulfillment
of the requirements for the award of the degree of BACHELOR OF ENGINEERING in
COMPUTER SCIENCE AND ENGINEERING at INEURON.AI Pvt. Ltd.

Internship CoordinatorProfessor & HeadDepartment of CSEDepartment of CSE

Internal Examiner External Examiner

Place:

Date:

21ETIT510

# INDUSTRIAL TRAINING / RURAL INTERNSHIP/ INNOVATION / ENTREPRENEURSHIP

L	TR	S	C
0	1	2	4

Four weeks during the summer vacation after the IV semester / VI semester of the programme COURSE OBJECTIVES:

- To expose the students to understand technical and professional skill requirements in IT industries.
- To impart professional skills for solving problems in industries.
- To train the students to design innovative solutions for a problem.
- To motivate the students to become an Entrepreneur.
- To develop communication and technical report writing skill.

The students will work for two periods per week guided by student counselor. They will be asked to present a seminar of not less than 15 minutes and not more than 30 minutes on any technical topic of student's choice related to Computer Science and Engineering and to engage in discussion with audience. They will defend their presentation. A brief copy of their presentation also should be submitted. Evaluation will be done by the student counselor based on the technical presentation, the report and on the interaction shown during the seminar.

The students will individually undertake a training program in reputed concerns in the field of Computer Science and Engineering during summer vacation for a minimum stipulated period of four weeks. At the end of training the student has to submit the detailed report on the training undertaken within ten days from the commencement of the seventh semester. The student will be evaluated by a team of staff members nominated by the Head of the Department through a viva-voce examination.

# **COURSE OUTCOMES:**

At the end of this course, the students will be able to

- 1. Understand the day-to-day job in IT industries, and technical and professional skills needed for an industry.
- 2. Develop and refine technical and professional skills through hands-on work experience.
- 3. Design an innovative solution for an Industry requirement by applying the knowledge learned from industry and in academics.
- 4. Develop a startup for product or services based on the people or industry requirements.

		Map	ping of	Course	e Outco	mes w	ith Prog	gramm	e Outco	omes		
	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CO1	1	1	-	-	-	_	_	-	-	1	ı	ı
CO2	1	2	2	-	-	-	_	-	-	-	-	3
CO3	1	-	2	1	2	-	_	-	-	-	-	-
CO4	1	-	_	-	-	_	_	-	2	-	2	1
CO5	1	-	-	-	2	-	_	-	-	3	-	-

# **Internship Completion Certificate**



# iNeuron Intelligence Pvt Ltd

17th Floor Tower A, Brigade Signature Towers, Sannatammanahalli, Bengaluru, Karnataka -562129.

#### Internship Experience Letter

DATE: 15th July 2023

#### TO WHOM IT MAY CONCERN

This is to certify that Mr/Ms/Mrs Sivaa Ganesh S has successfully completed internship program from 9th June 2023 to 15th July 2023 in **Campus Placement** at INEURON INTELLIGENCE PRIVATE LIMITED. During their internship programme with us, they demonstrated exceptional skills with a self-motivated attitude to learn new things and implement them end to end with all of our mentioned industrial standards. Their performance was excellent and was able to complete the project successfully on time.

We wish them all the best for future endeavours.

Regards,

Sudhanshu Kumar

Justin

CEO & Chief Al Engineer at iNeuron.ai

# **Internship Objectives**

Internships are generally thought to be reserved for college students looking to gain experience in a particular field. However, a wide array of people can benefit from training internships in order to receive real-world experience and develop their skills.

Internships are utilized in a number of different career fields, including architecture, engineering, healthcare, economics, advertising, and many more.

Some internships are used to allow individuals to perform scientific research, while others are specifically designed to allow people to gain first-hand experience working.

Utilizing internships is a great way to build your resume and develop skills that can be emphasized in your resume for future jobs.

Internship learning objectives should be developed along four dimensions as follows:

- 1. **Skill development:** Learning and improving skills such as writing, verbal communication, research, technology, teamwork, and leadership. It is the development of these skills that often represent the major benefits of an internship.
- 2. **Understanding Real-World Application:** Understanding the workplace, operating procedures, the department/company, its products, and other organizational concepts. In addition, this would include knowledge added to existing classroom knowledge, such as new applications or new skills.
- 3. **Career Awareness:** Internships often provide the opportunity to take a peek at what working for a company or in an industry would be like. Objectives could include learning about career positions and occupations, along with the qualities and training required to obtain those positions.
- 4. **Personal Development:** One of the major benefits of an internship is how it helps you to develop self-confidence, assertiveness, and basic work habits

# **Table of Contents**

S.No	Contents	Page No.
1	Introduction	1
2	Abstract	2
3	Introduction to Data Science and Machine Learning	3
4	Problem Statements & Proposed Solutions	4
5	Software Requirements	4
6	Project Scope & Methodologies	5
7	Screenshots	6
8	Code	15
9	References	27
10	Conclusion	28

# Introduction

In the realm of campus placement prediction, the focus lies on utilizing advanced technology and smart algorithms to anticipate where a student is likely to be placed. My proficiency in AI and Machine Learning has equipped me with powerful tools that can be transformative in this context.

However, bridging the gap between theoretical knowledge and practical application is key. This is where campus placement prediction comes into play. It serves as the platform where these sophisticated concepts can be employed to address a real-world challenge that impacts a multitude of students.

During my internship, I delved into three distinct projects: a Campus Placement Estimator, a tool to discern enrollment trends, and a predictor for job demand. Each undertaking presented unique hurdles and bestowed invaluable insights, ultimately enhancing my prowess in predicting campus placements.

In this report, I will provide an in-depth account of these projects, detailing the methodologies I employed and the lessons I garnered. I aim to showcase my fervor for employing technology to address tangible problems, particularly in the domain of campus placement prediction. The objective is to effect a positive change in our interconnected educational landscape. So, let's delve into each project and witness how I'm leaving a mark in this dynamic field.

# **Abstract**

- The "Enhancing Campus Placement Process through AI-driven Analytics and Personalized Recommendations" project stands at the forefront of transforming the traditional campus placement process. By leveraging advanced Artificial Intelligence (AI) techniques, this initiative addresses the persistent challenges faced by both students and recruiters in aligning skill sets and preferences effectively.
- This innovative system adopts a comprehensive approach, seamlessly integrating data analytics, machine learning algorithms, and natural language processing techniques. It commences with meticulous data collection, encompassing a wide array of student-centric attributes such as academic records, skill assessments, and career aspirations. Concurrently, it compiles in-depth job profiles and skill requirements from potential employers, forming a comprehensive database.
- One of the distinguishing features of this project is its adaptability. The system is engineered to
  continuously refine its recommendations through iterative machine learning processes. This
  involves the incorporation of invaluable feedback from both students and recruiters, ensuring a
  progressively accurate and efficient matching process over time.
- The anticipated outcomes of this endeavor are far-reaching. It aims to create a campus placement process that is not only streamlined but highly effective, offering substantial benefits to both students and recruiters. Students gain access to a thoughtfully curated selection of opportunities that resonate with their career aspirations and individual skill sets. Recruiters, in turn, receive a pool of candidates who not only possess the requisite skills but also exhibit attributes that closely align with the specific needs of their organizations.
- In conclusion, the "Enhancing Campus Placement Process through AI-driven Analytics and Personalized Recommendations" project embodies a significant stride towards optimizing the campus placement ecosystem. By harnessing the potential of AI and capitalizing on data-driven insights, this endeavor holds the promise of reshaping the student-recruiter connection within the domain of campus placements. It envisions a future where the placement process is characterized by precision, efficiency, and mutual satisfaction for both students and recruiters alike. This project represents a pioneering leap towards a more effective and equitable campus placement landscape.

# **Introduction to Data Science and Machine learning**

Data science and artificial intelligence (AI) play a pivotal role in forecasting **campus placements** for **educational institutions.** These technologies empower institutions to analyze diverse sets of data, extract valuable insights, and make informed decisions to enhance placement strategies. Below is a concise overview of how data science and AI are utilized in predicting campus placements:

- 1. **Data Collection:** Educational institutions gather a wide array of data pertaining to their students, including academic records, extracurricular activities, internship experiences, and career preferences.
- Data Preprocessing: Prior to applying AI techniques, the collected data needs to be cleaned, transformed, and prepared for analysis. This involves handling missing values, outliers, and standardizing the data.
- 3. **Feature Engineering:** Identifying and creating relevant features (variables) that have an impact on placements is crucial. This step lays the foundation for building accurate prediction models.
- 4. **Data Analysis:** Data scientists employ statistical and exploratory data analysis techniques to discern patterns, correlations, and trends in the data. This aids in identifying factors that influence campus placements.
- 5. **Machine Learning Models:** AI techniques, particularly machine learning, are employed to construct predictive models. Classification algorithms, such as logistic regression, support vector machines, and decision trees, are commonly used for placement prediction.
- 6. **Model Training**: The collected data is divided into training and testing sets. Machine learning models are trained on the training data to understand the relationships between the features and placement outcomes.
- 7. **Model Evaluation:** The trained models are evaluated using the testing data to assess their predictive accuracy. Metrics like accuracy, precision, recall, and F1-score are often used to measure model performance.
- 8. **Hyperparameter Tuning:** Data scientists fine-tune the model's hyperparameters to enhance its performance. This iterative process aids in optimizing the model's predictive capabilities.
- 9. **Deployment:** Once a satisfactory model is developed, it can be deployed in a real-world setting to predict campus placements. This can be integrated into the institution's placement strategy.

# **Problem Statements**

to predict whether the student will be recruited in campus placements or not based on the available factors in the dataset.

# **Proposed Solutions**

A Web UI that predicts the suitability of a candidate for a particular campus placement opportunity based on their provided information, which it has been trained on, is a valuable tool for guiding students towards optimal career choices. It offers educational institutions and students a powerful solution for making informed placement decisions effectively.

# **Software Requirements:**

Campus placement prediction involves the application of various machine learning and statistical modeling techniques. To implement these techniques effectively, you can use a combination of programming languages and libraries specifically designed for data analysis and machine learning. Some commonly used tools for campus placement prediction include:

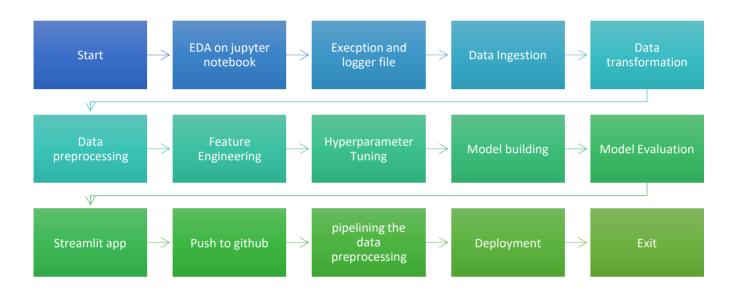
- **Python**: Python is a popular programming language for data science and machine learning. It offers a wide range of libraries and frameworks, such as NumPy, pandas, scikit-learn, and TensorFlow, that facilitate data manipulation, model building, and evaluation.
- **Jupyter Notebook**: Jupyter Notebook is an interactive development environment that allows you to combine code, visualizations, and explanatory text in a single document. It's commonly used for exploratory data analysis and model prototyping.
- **scikit-learn**: This is a powerful machine learning library for Python that provides various algorithms for regression, classification, clustering, and more. It's widely used for building predictive models, including sales price prediction.
- **Matplotlib**: Matplotlib is a widely-used data visualization library in Python. It provides a flexible and comprehensive set of functions to create various types of static, interactive, and animated plots.
- **Seaborn**: Seaborn is a Python data visualization library built on top of Matplotlib. It provides a high-level interface for creating aesthetically pleasing statistical graphics.

# **Project Scope And Methodologies**

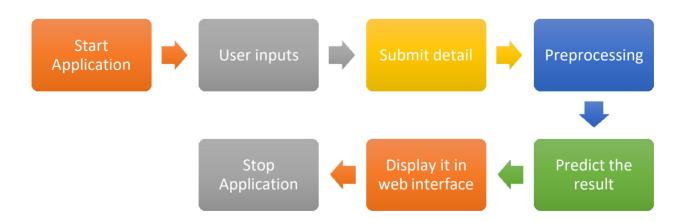
# Scope

The objective is to develop an effective and precise predictive solution that can forecast the suitability of candidates for various campus placement opportunities offered by educational institutions. This task holds immense significance for both students and institutions, as it enables them to make well-informed decisions regarding career paths, skill development, and placement strategies.

#### **Process Flow**



# **Deployment Process**



# **Source Code**

# **Exploratory Data Analysis:**

# **Campus placements Prediction**

#### 1) Problem statement

• to predict whether the student will be recruited in campus placements or not based on the available factors in the dataset.

#### 2) Data collection

- Dataset Source <a href="https://www.kaggle.com/competitions/ml-with-python-course-project/data">https://www.kaggle.com/competitions/ml-with-python-course-project/data</a>
  - . The train data consists of 15 column and 215 rows.
  - . The test data consist of 3 columns and 43 rows.

#### 2.1 Import Data and Required Packages

Importing Pandas, Numpy, Matplotlib, Seaborn and Warings Library.

```
In [1]:
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

#### Import the CSV Data as Pandas DataFrame

#### Show Top 5 Records

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	0	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	0	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	0	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	0	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	0	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

#### 2.2 Dataset information

- sl no: anonymous id unique to a given employee
- · gender: employee gender
- ssc\_p: SSC is Secondary School Certificate (Class 10th). ssc\_p is the percentage of marks secured in Class 10th.
- · ssc b: SSC Board. Binary feature.
- · hsc p: HSC is Higher Secondary Certificate (Class 12th), hsc p is the percentage of marks secured in Class 12th.
- · hsc b: HSC Board. Binary feature.
- . hsc s: HSC Subject. Feature with three categories.
- degree p: percentage of marks secured while acquiring the degree.
- degree t: branch in which the degree was acquired. Feature with three categories.
- . workex: Whether the employee has some work experience or not. Binary feature.
- etest\_p: percentage of marks secured in the placement exam.
- · specialisation: the specialization that an employee has. Binary feature.
- . mba\_p: percentage of marks secured by an employee while doing his MBA.
- . status: whether the student was placed or not. Binary Feature. Target variable.
- · salary: annual compensation at which an employee was hired.

#### 3. Data Checks to perform

- · Check Missing values
- · Check Duplicates
- · Check data type
- · Check the number of unique values of each column
- · Check statistics of data set
- · Check various categories present in the different categorical column

#### 3.1 Check Missing values

```
In [16]: df.isna().sum()
Out[16]: sl no
       gender
                      A
                    0
       ssc_p
       ssc b
       hsc_p
       hsc_b
                     0
       hsc_s
       degree_p
                     0
       degree_t
       workex
                     0
       etest p
       specialisation 0
       mba p
                     0
       status
       salary
                     67
       dtype: int64
```

#### Observation:

. There are 67 missing values in salary depicts that 67 out of 215 has not been placed

In [17]: df Out[17]: sl\_no gender ssc\_p ssc\_b hsc\_p hsc\_b hsc\_s degree\_p degree\_t workex etest\_p specialisation mba\_p status salary 0 0 67.00 Others 91.00 Others Commerce Placed 270000.0 58 00 Sci&Tech No 55.0 Mkt&HR 58 80 1 77.48 Sci&Tech Placed 200000.0 0 79.33 Central 78.33 Others 86.5 Mkt&Fin 66.28 Science Yes 2 0 65.00 Central 68.00 Central Arts 64.00 Comm&Mgmt 75.0 Mkt&Fin 57.80 Placed 250000.0 No Sci&Tech 3 4 0 56.00 Central 52.00 Central Science 52.00 No 66.0 Mkt&HR 59.43 Not Placed NaN 4 0 85.80 Central 73.60 Central Commerce 73.30 Comm&Mgmt 96.8 Mkt&Fin 55.50 Placed 425000.0 5 No 211 0 80.60 Others 82.00 Others Commerce 77.60 Comm&Mgmt 91.0 Mkt&Fin 74.49 Placed 400000.0 210 No 211 212 0 58.00 Others 60.00 Others 72.00 Sci&Tech 74.0 Mkt&Fin 53.62 Placed 275000.0 Science No 59.0 212 213 0 67.00 Others 67.00 Others Commerce 73.00 Comm&Mgmt Yes Mkt&Fin 69.72 Placed 295000.0 Placed 204000.0 213 214 1 74.00 Others 66.00 Others Commerce 58.00 Comm&Mgmt 70.0 Mkt&HR 60.23 No 214 215 0 62.00 Central 58.00 Others Science 53.00 Comm&Mgmt No 89.0 Mkt&HR 60.22 Not Placed NaN

215 rows x 15 columns

#### 3.2 Check Duplicates

```
In [18]: df.duplicated().sum()
```

Out[18]: 0

There are no duplicates values in the data set

#### 3.3 Check data types

```
In [19]: df.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215 entries, 0 to 214
Data columns (total 15 columns):

ata	columns (total	15 CO.	Tumns	5):	
#	Column	Non-	Null	Count	Dtype
0	sl_no	215	non-r	null	int64
1	gender	215	non-r	null	int64
2	ssc_p	215	non-r	null	float64
3	ssc_b	215	non-r	null	object
4	hsc_p	215	non-r	null	float64
5	hsc_b	215	non-r	null	object
6	hsc_s	215	non-r	null	object
7	degree_p	215	non-r	null	float64
8	degree_t	215	non-r	null	object
9	workex	215	non-r	null	object
10	etest_p	215	non-r	null	float64
11	specialisation	215	non-r	null	object
12	mba_p	215	non-r	null	float64
13	status	215	non-r	null	object
14	salary	148	non-r	null	float64
tun	oc: float64(6)	int64	(2)	object	(7)

dtypes: float64(6), int64(2), object(7)

memory usage: 25.3+ KB

# 3.4 Checking the number of unique values of each column

]: df.nunique()			
]: sl_no	215		
gender	2		
ssc_p	103		
ssc_b	2		
hsc_p	2 97		
hsc_b	2		
hsc_s	3		
degree_p	3 89 3 2		
degree_t	3		
workex	2		
etest_p	100		
specialisation	2		
mba_p	205		
status	205 2 45		
salary	45		
dtype: int64			

#### 3.5 Check statistics of data set

In [21]:	df.des	cribe()							
Out[21]:		sl_no	gender	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
	count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
	mean	108.000000	0.353488	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
	std	62.209324	0.479168	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420

count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	0.353488	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	0.479168	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	0.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	0.000000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	0.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	1.000000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	1.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

#### Observation

The given data is perfect with no outliers.

#### 3.6 Exploring Data

lf	.head(	)													
	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etest_p	specialisation	mba_p	status	salary
0	1	0	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	0	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	0	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	0	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	0	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0

```
In [24]: # define numerical & categorical columns
         numeric features = [feature for feature in df.columns if df[feature].dtype != '0']
        categorical_features = [feature for feature in df.columns if df[feature].dtype == '0']
         print('We have {} numerical features : {}'.format(len(numeric_features), numeric_features))
         print('\nWe have {} categorical features : {}'.format(len(categorical_features), categorical_features))
         We have 8 numerical features : ['sl_no', 'gender', 'ssc_p', 'hsc_p', 'degree_p', 'etest_p', 'mba_p', 'salary']
         We have 7 categorical features : ['ssc b', 'hsc b', 'hsc s', 'degree t', 'workex', 'specialisation', 'status']
In [25]: for feature in categorical_features:
             print(f'unique values in "{feature}" column: ', df[feature].unique())
             print()
         unique values in "ssc_b" column: ['Others' 'Central']
         unique values in "hsc_b" column: ['Others' 'Central']
         unique values in "hsc_s" column: ['Commerce' 'Science' 'Arts']
         unique values in "degree_t" column: ['Sci&Tech' 'Comm&Mgmt' 'Others']
         unique values in "workex" column: ['No' 'Yes']
         unique values in "specialisation" column: ['Mkt&HR' 'Mkt&Fin']
         unique values in "status" column: ['Placed' 'Not Placed']
```

#### Observations

- . The data look perfect with no mistaken records.
- · the naming convention maybe improved slightly for better understanding.

#### **Model Training:**

#### 1.1 Import Data and Required Packages

Importing Pandas, Numpy, Matplotlib, Seaborn and Warings Library.

```
In [1]: # Basic Import
       import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        # Modellina
        from sklearn.preprocessing import StandardScaler
        from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
        from sklearn.svm import SVC
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
        from sklearn.model_selection import RandomizedSearchCV
        from catboost import CatBoostClassifier
        from xgboost import XGBClassifier
        import warnings
```

#### Import the CSV Data as Pandas DataFrame

```
In [2]: df = pd.read_csv('data/train.csv')
```

#### Show Top 5 Records

```
In [4]: df.head()
 Out[4]:
             sl_no gender ssc_p ssc_b hsc_p hsc_b
                                                        hsc s degree p
                                                                          degree t workex etest p specialisation mba p
                                                                                                                        status
                                                                                                                                salary
                       0 67.00 Others 91.00 Others Commerce
                                                                 58.00
                                                                          Sci&Tech
                                                                                            55.0
                                                                                                      Mkt&HR 58.80
                                                                                                                        Placed 270000.0
                        0 79.33 Central 78.33 Others
                                                       Science
                                                                 77 48
                                                                          Sci&Tech
                                                                                      Yes
                                                                                             86.5
                                                                                                      Mkt&Fin 66.28
                                                                                                                        Placed 200000 0
                       0 65.00 Central 68.00 Central
                                                         Arts
                                                                 64.00 Comm&Mgmt
                                                                                      No
                                                                                            75.0
                                                                                                      Mkt&Fin 57.80
                                                                                                                        Placed 250000.0
                4
                       0 56 00 Central 52 00 Central
                                                                 52 00
                                                                          Sci&Tech
                                                                                            66.0
                                                                                                      Mkt&HR 59 43 Not Placed
                                                                                                                                  NaN
           3
                                                      Science
                                                                                      No
                       0 85.80 Central 73.60 Central Commerce 73.30 Comm&Mgmt
                                                                                                      Mkt&Fin 55.50
                                                                                                                        Placed 425000.0
 In [5]: df.drop('sl_no', axis=1, inplace=True)
 In [6]: df['salary'] = df['salary'].fillna(0)
 In [7]: df['status'] = df['status'].replace({'Placed': 1, 'Not Placed': 0})
 In [8]: df.drop('salary', axis=1, inplace=True)
 In [9]: df.head()
 Out[9]:
                                                                    degree_t workex etest_p specialisation mba_p status
             gender ssc p ssc b hsc p hsc b
                                                  hsc s degree p
           0 0 67.00 Others 91.00 Others Commerce
                                                           58.00 Sci&Tech
                                                                                No
                                                                                      55.0
                                                                                                Mkt&HR 58.80
                  0 79.33 Central 78.33 Others
                                                Science
                                                           77.48
                                                                    Sci&Tech
                                                                                Yes
                                                                                      86.5
                                                                                                Mkt&Fin
                                                                                                         66.28
                                                  Arts
           2
                 0 65.00 Central 68.00 Central
                                                           64.00 Comm&Mgmt
                                                                                No
                                                                                      75.0
                                                                                                Mkt&Fin
                                                                                                         57.80
                                                           52.00
                                                                    Sci&Tech
                                                                                      66.0
                                                                                                Mkt&HR
           3
                 0 56.00 Central 52.00 Central Science
                                                                                No
                                                                                                         59.43
                                                                                                                   0
                                                                                                Mkt&Fin
                 0 85.80 Central 73.60 Central Commerce 73.30 Comm&Mgmt
                                                                               No
                                                                                      96.8
                                                                                                         55.50
In [10]: X = df.drop(columns=['status'], axis=1)
         y = df['status']
In [11]: # Create Column Transformer with 3 types of transformers
         num_features = X.select_dtypes(exclude="object").columns
cat_features = X.select_dtypes(include="object").columns
         from sklearn.preprocessing import OneHotEncoder, StandardScaler
         from sklearn.compose import ColumnTransformer
         numeric transformer = StandardScaler()
         oh_transformer = OneHotEncoder()
         preprocessor = ColumnTransformer(
                  ("OneHotEncoder", oh_transformer, cat_features),
                   ("StandardScaler", numeric_transformer, num_features),
In [12]: X = preprocessor.fit_transform(X)
In [13]: X.shape
Out[13]: (215, 20)
         Preparing X and Y variables
In [14]: # separate dataset into train and test
          from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=42)
         X_train.shape, X_test.shape
Out[14]: ((172, 20), (43, 20))
```

#### Create an Evaluate Function to give all metrics after model Training

In [15]: def evaluate\_model(true, predicted):

```
mae = mean_absolute_error(true, predicted)
                mse = mean_squared_error(true, predicted)
                rmse = np.sqrt(mean_squared_error(true, predicted))
                r2 square = r2 score(true, predicted)
                return mae, rmse, r2_square
In [22]: models = {
    "Logistic Regression": LogisticRegression(),
    "K-Neighbors Classifier": KNeighborsClassifier(),
    "Decision Tree Classifier": DecisionTreeClassifier(),
                 "Random Forest Classifier": RandomForestClassifier(),
                "SVM Classifier": SVC(),
                "XGBoost Classifier": XGBClassifier(),
"CatBoost Classifier": CatBoostClassifier(verbose=False),
"AdaBoost Classifier": AdaBoostClassifier()
            model_list = []
            accuracy_list = []
            for model_name, model in models.items():
                model.fit(X_train, y_train) # Train model
                # Make predictions
                y_train_pred = model.predict(X_train)
                y_test_pred = model.predict(X_test)
                # Evaluate Train and Test dataset
                accuracy_train = accuracy_score(y_train, y_train_pred)
                accuracy_test = accuracy_score(y_test, y_test_pred)
                print(model_name)
                model_list.append(model_name)
                print('\nModel performance for Training set')
print("- Accuracy: {:.4f}".format(accuracy_train))
                print('Model performance for Test set')
                print("- Accuracy: {:.4f}".format(accuracy_test))
                accuracy_list.append(accuracy_test)
                print('=' * 35)
print('\n')
            # Now you can access the model names and their accuracy scores in model list and accuracy list, respectively.
         Logistic Regression
```

```
Decision Tree Classifier
Model performance for Training set
- Accuracy: 1.0000
Model performance for Test set
- Accuracy: 0.8372
-----
Random Forest Classifier
Model performance for Training set
- Accuracy: 1.0000
Model performance for Test set
- Accuracy: 0.8372
SVM Classifier
Model performance for Training set
- Accuracy: 0.9360
Model performance for Test set
- Accuracy: 0.8140
XGBoost Classifier
Model performance for Training set
- Accuracy: 1.0000
_____
Model performance for Test set
- Accuracy: 0.8372
CatBoost Classifier
Model performance for Training set
- Accuracy: 0.9942
Model performance for Test set
- Accuracy: 0.7907
AdaBoost Classifier
Model performance for Training set
- Accuracy: 1.0000
Model performance for Test set
- Accuracy: 0.8605
_____
Results
```

#### Out[19]:

```
        Model Name
        R2_Score

        0
        Logistic Regression
        0.883721

        7
        AdaBoost Classifier
        0.860465

        2
        Decision Tree Classifier
        0.837209

        5
        XGBoost Classifier
        0.837209

        4
        SVM Classifier
        0.813953

        1
        K-Neighbors Classifier
        0.790698

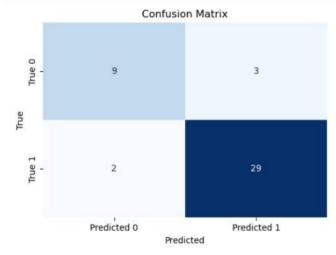
        3
        Random Forest Classifier
        0.790698

        6
        CatBoost Classifier
        0.790698
```

```
In [20]: log_model = LogisticRegression()
    log_model = log_model.fit(X_train, y_train)
    y_pred = log_model.predict(X_test)
    score = accuracy_score(y_test, y_pred)*100
    print(" Accuracy of the model is %.2f" %score)
```

Accuracy of the model is 88.37

#### Plot y\_pred and y\_test



#### Observation

model predicted with 5 wrong prediction.

#### **Data\_Ingestion.py**

```
import os
import sys
sys.path.append(r'D:\Programs\Machine learning projects\Campus placement pre
diction')
from dataclasses import dataclass
import pandas as pd
from sklearn.model selection import train test split
from src.Components.data transformation import DataTransformationConfig
from src.Components.data transformation import DataTransformation
from src.Components.model trainer import ModelTrainer
from src.Components.model trainer import ModelTrainerConfig
from src.exception import CustomException
from src.logger import logging
@dataclass
class DataIngestionConfig:
    train data_path: str= os.path.join('artifacts', 'train_data.csv')
    test_data_path: str= os.path.join('artifacts', 'test_data.csv')
    raw_data_path: str= os.path.join('artifacts', 'data.csv')
class DataIngestion:
    def init (self):
        self.ingestion config = DataIngestionConfig()
    def initiate data ingestion(self):
        logging.info('Entered data ingestion')
        try:
            df = pd.read csv('Notebooks\data\Train clean.csv')
            logging.info('Reading the dataset.')
            os.makedirs(os.path.dirname(self.ingestion config.test data path
), exist_ok=True)
            df.to csv(self.ingestion config.raw data path, index=False,
header=True)
            logging.info("Train test split initiated")
            train set, test set = train test split(df, test size=0.2,
random state=42)
```

```
train set.to csv(self.ingestion config.train data path,
index=False, header=True)
            test set.to csv(self.ingestion config.test data path,
index=False, header=True)
            logging.info('Ingestion of data is completed.')
            return (
                self.ingestion config.train data path,
                self.ingestion config.test data path
            )
        except Exception as e:
            raise CustomException(e, sys)
if__name__ == '__main__':
    obj = DataIngestion()
    train data, test data = obj.initiate data ingestion()
    data transformation = DataTransformation()
    train_arr, test_arr, _=
data transformation.initiate data transformation(train data, test data)
    model = ModelTrainer()
    print(model.initiate model trainer(train arr, test arr))
Data_Transformation.py
import sys
from dataclasses import dataclass
import numpy as np
import pandas as pd
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder,StandardScaler
from src.exception import CustomException
from src.logger import logging
import os
from src.utils import save_object
@dataclass
```

```
class DataTransformationConfig:
    preprocessor obj file path=os.path.join('artifacts', "preprocessor.pkl")
class DataTransformation:
    def init (self):
        self.data transformation config=DataTransformationConfig()
    def get_data_transformer_object(self):
        This function is responsible for data transformation
        . . .
        try:
            numerical_columns = ['gender', 'ssc_p', 'hsc_p', 'degree_p',
'etest_p', 'mba_p']
            categorical columns = ['ssc b', 'hsc b', 'hsc s', 'degree t',
'workex', 'specialisation']
            num_pipeline= Pipeline(
                steps=[
                ("imputer", SimpleImputer(strategy="median")),
                ("scaler",StandardScaler())
                ]
            )
            cat pipeline=Pipeline(
                steps=[
                ("imputer", SimpleImputer(strategy="most_frequent")),
                ("one hot encoder", OneHotEncoder(handle unknown="ignore")),
                ("scaler",StandardScaler(with_mean=False))
            )
            logging.info(f"Categorical columns: {categorical_columns}")
            logging.info(f"Numerical columns: {numerical_columns}")
            preprocessor=ColumnTransformer(
                ("num pipeline", num pipeline, numerical columns),
                ("cat_pipelines", cat_pipeline, categorical_columns)
```

```
]
            )
            return preprocessor
        except Exception as e:
            raise CustomException(e,sys)
    def initiate data transformation(self,train path,test path):
        try:
            train df=pd.read csv(train path)
            test_df=pd.read_csv(test_path)
            logging.info("Read train and test data completed")
            logging.info("Obtaining preprocessing object")
            preprocessing_obj=self.get_data_transformer_object()
            target_column_name="status"
            input_feature_train_df=train_df.drop(columns=[target_column_name
],axis=1)
            target_feature_train_df=train_df[target_column_name].values
            input feature test df=test df.drop(columns=[target column name],
axis=1)
            target feature test df=test df[target column name].values
            logging.info(
                f"Applying preprocessing object on training dataframe and
testing dataframe."
            )
            print(input_feature_train_df.columns)
            print(input_feature_test_df.columns)
            input_feature_train_arr=preprocessing_obj.fit_transform(input_fe
ature_train_df)
```

```
input feature test arr=preprocessing obj.transform(input feature
test df)
            train arr = np.c [input feature train arr,
np.array(target feature train df)]
            test_arr = np.c_[input_feature_test_arr,
np.array(target feature test df)]
            logging.info(f"Array concatenated sucessfully.")
            save object(
                file_path=self.data_transformation_config.preprocessor_obj_f
ile path,
                obj=preprocessing_obj
            )
            logging.info(f"Saved preprocessing object.")
            return (
                train arr,
                test_arr,
                self.data transformation config.preprocessor obj file path,
        except Exception as e:
            raise CustomException(e,sys)
Model_Trainer.py
import os
import sys
from dataclasses import dataclass
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy score, precision score, recall score,
f1_score, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.svm import SVC
```

from sklearn.linear model import LogisticRegression

```
from sklearn.metrics import accuracy score, precision score, recall score,
f1 score, confusion matrix
from sklearn.model selection import RandomizedSearchCV
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
from src.exception import CustomException
from src.logger import logging
from src.utils import save object, evaluate model
@dataclass
class ModelTrainerConfig:
    trained model file path: str= os.path.join('artifacts', 'model.pkl')
class ModelTrainer:
    def init (self) -> None:
        self.model_trainer_config = ModelTrainerConfig()
    def initiate_model_trainer(self, train_array, test_array):
        try:
            logging.info('Entered model trainer. Splitting train and test
dataset.')
            x_train, y_train, x_test, y_test = (
                train_array[:, :-1],
                train_array[:, -1],
                test array[:, :-1],
                test_array[:, -1]
            )
            models = {
                        "Logistic Regression": LogisticRegression(),
                        "K-Neighbors Classifier": KNeighborsClassifier(),
                        "Decision Tree Classifier":
DecisionTreeClassifier(),
                        "Random Forest Classifier":
RandomForestClassifier(),
                        "SVM Classifier": SVC(),
                        "XGBoost Classifier": XGBClassifier(),
                        "CatBoost Classifier":
CatBoostClassifier(verbose=False),
                        "AdaBoost Classifier": AdaBoostClassifier()
                    }
```

```
model report= evaluate model(X train=x train, y train= y train,
X test= x test, y test= y test, models= models)
            best model score = max(sorted(model report.values()))
            best model name =
list(model report.keys())[list(model report.values()).index(best model score
)1
            best model = models[best model name]
            logging.info('Best model found.')
            save object(
                file path=
self.model trainer config.trained model file path,
                obj=best model
            )
            predicted=best model.predict(x test)
            acc_value = accuracy_score(y_test, predicted)
            return f"Model name: {best model name}\nr2 score: {acc value}"
        except Exception as e:
            raise CustomException(e, sys)
Exception.py
import sys
from src.logger import logging
def error_message_detail(error, error_detail: sys):
    _, _, exc_tb = error_detail.exc_info()
    filename = exc_tb.tb_frame.f_code.co_filename
    lineno = exc_tb.tb_lineno
    error message = "Error occured in python script \nname : {0}\nLine
number: {1}\nError message: {2}".format(
        filename, lineno, error
    )
    return error message
class CustomException(Exception):
    def_init_(self, error message, error detail: sys):
        super(). init (error message)
```

```
self.error_message = error_message_detail(error_message,
error detail)
    def str (self) -> str:
        return self.error_message
if name == " main ":
    try:
        a = 1/0
    except Exception as e:
        logging.exception(e)
        raise CustomException(e, sys)
Utils.py
import os
import pickle
import sys
from sklearn.metrics import accuracy score
from sklearn.model selection import GridSearchCV
from src.exception import CustomException
from src.logger import logging
def save object(file path, obj):
    try:
        dir path= os.path.dirname(file path)
        os.makedirs(dir_path, exist_ok=True)
        with open(file_path, "wb") as file_obj:
            pickle.dump(obj, file_obj)
    except Exception as e:
        raise CustomException(e, sys)
def evaluate model(X train, y train, X test, y test, models):
    try:
        report={}
        for i in range(len(list(models))):
            model = list(models.values())[i]
            logging.info(f"In the model: {model}")
```

```
model.fit(X train, y train)
            y test pred = model.predict(X test)
            test model score = accuracy score(y test, y test pred)
            report[list(models.keys())[i]] = test model score
            logging.info(f'Completed: {model}')
        return report
    except Exception as e:
        raise CustomException(e, sys)
def load model(file path):
    try:
        with open(file path, 'rb') as obj:
            return pickle.load(obj)
    except Exception as e:
        raise CustomException(e, sys)
Logger.py
import logging
import os
from datetime import datetime
LOG FILE = f"{datetime.now().strftime('%m %d %Y %H %M %S')}.log"
logs path = os.path.join(os.getcwd(), "logs", LOG FILE)
os.makedirs(logs path, exist ok=True) # To append file after the folder is
created
LOG FILE PATH = os.path.join(logs path, LOG FILE)
logging.basicConfig(
    filename= LOG FILE PATH,
    format= "[ %(asctime)s ] %(lineno)d %(name)s - %(levelname)s -
%(message)s",
    level= logging.INFO
)
if_name__== "__main__":
   logging.info("Test logging started.")
```

# App.py

```
import streamlit as st
import pandas as pd
import numpy as np
from src.pipelines.predict pipeline import PredictPipeline
st.write("""
# Campus prediction App.
This app predicts whether the student will be recruited in campus placements
or not based on the available factors in the dataset.
Data obtained from [Kaggle campus
placement](https://www.kaggle.com/competitions/ml-with-python-course-
project/data) """)
st.sidebar.header('User Input Features')
st.sidebar.markdown("""
 [Example CSV input file]() """)
# Collects user input features into dataframe
uploaded file = st.sidebar.file uploader("Upload your input CSV file",
type=["csv"])
if uploaded file is not None:
    input df = pd.read csv(uploaded file)
else:
    def user input features():
        GENDER = {'Male': 0, 'Female': 1}
        DEGREE_T = {'Science and Technology':'Sci&Tech', 'Commerece and
Management': 'Comm&Mgmt', 'Others': 'Others'}
        SPECIALISATION= {'Market and HR': 'Mkt&HR', 'Market and Finance':
'Mkt&Fin'}
        gender = st.sidebar.selectbox('Gender', ('Male', 'Female'))
        ssc_p = st.sidebar.slider('Percentage of Mark scored in 10th', 0,
100, 40)
        ssc b = st.sidebar.selectbox('Which Board Studied in 10th',
('Others', 'Central'))
        hsc p = st.sidebar.slider('Percentage of Mark scored in 12th', 0,
100, 40)
        hsc s = st.sidebar.selectbox('Which Stream Studied in 12th',
('Science', 'Commerce', 'Arts'))
```

```
hsc b = st.sidebar.selectbox('Which Board Studied in 12th',
('Others', 'Central'))
        degree p = st.sidebar.slider('Percentage of Mark scored while
completing Degree', 0, 100, 40)
        degree t = st.sidebar.selectbox('Branch in which Degree was
Acquired', ('Science and Technology', 'Commerece and Management', 'Others'))
        etest p = st.sidebar.slider('Percentage of Mark scored placement
exam', 0, 100, 40)
        workexp = st.sidebar.selectbox('Do you have prior work experience',
('yes', 'no'))
        mba p = st.sidebar.slider('Percentage of Mark scored in MBA', 0,
100, 40)
        speci = st.sidebar.selectbox('Do you have specialization in any of
the below criteria', ('Market and HR', 'Market and Finance'))
        gender 1 = GENDER[gender]
        degree t 1 = DEGREE T[degree t]
        speci 1 = SPECIALISATION[speci]
        data = {
                'gender': gender 1,
                'ssc_p': ssc_p,
                'ssc b': ssc b,
                'hsc_p': hsc_p,
                'hsc_s': hsc_s,
                'hsc b': hsc b,
                'degree_p': degree_p,
                'degree_t': degree_t_1,
                'etest p': etest p,
                'workex': workexp,
                'mba_p': mba_p,
                'specialisation': speci 1
        features = pd.DataFrame(data, index=[0])
        return features
    input_df = user_input_features()
st.subheader('User Input features')
if uploaded file is not None:
    st.write(input df)
else:
    st.write('Awaiting CSV file to be uploaded. Currently using example
input parameters (shown below).')
```

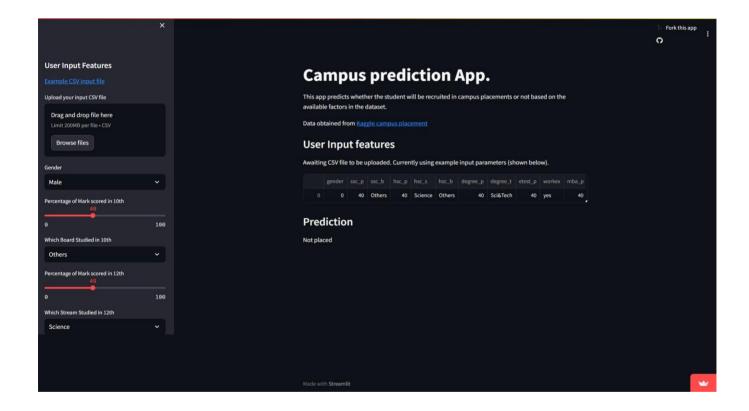
```
st.write(input_df)

predict_pipeline = PredictPipeline()

predicted_value= predict_pipeline.predict(input_df)

st.subheader('Prediction')
st.write('will be Placed' if predicted_value ==1 else 'Not placed')
```

# Web page Output:



# References

1. Programming Language -

**Python:** Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum-<a href="https://realpython.com/">https://realpython.com/</a>

#### 2. IDE -

**Jupyter Notebook:** The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text- https://jupyter.org/

- **3. Machine Learning :** What is machine learning? Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy <a href="https://www.geeksforgeeks.org/machine-learning/">https://www.geeksforgeeks.org/machine-learning/</a>
- **4. Streamlit**: Streamlit is an open-source app framework for Machine Learning and Data Science teams. Create beautiful web apps in minutes. <a href="https://docs.streamlit.io/">https://docs.streamlit.io/</a>
- **5. GitHub**: GitHub, Inc. is a platform and cloud-based service for software development and version control using Git, allowing developers to store and manage their code. <a href="https://docs.github.com">https://docs.github.com</a>
- **6. Referred GitHub repository :** Referred project for hyper parameter tuning and folder structure https://github.com/krishnaik06/mlproject
- 7. Referred YouTube playlist: Referred playlist to learn about the concept of modularizing code, data pipelining and various components <a href="https://youtube.com/playlist?list=PLZoTAELRMXVPS-dOaVbAux22vzqdgoGhG&si=RAdzHEaiQTF8LeSV">https://youtube.com/playlist?list=PLZoTAELRMXVPS-dOaVbAux22vzqdgoGhG&si=RAdzHEaiQTF8LeSV</a>

# **Conclusion**

- In conclusion, this data science project has provided valuable insights into predicting campus placements based on various factors such as educational performance, board, gender, work experience, and specialization.
- Moreover, our data science workflow demonstrated the significance of data preprocessing, feature engineering, and model selection in achieving accurate predictions. We addressed challenges such as handling missing data, which played a crucial role in achieving the project's objectives.
- Our analysis covered a range of features including SSC and HSC percentages, board types, degree marks, work experience, placement exam scores, MBA performance, and specialization choices. These factors were key in determining whether a student would be placed or not.
- As the field of data science continues to evolve, there are boundless opportunities for further exploration and discovery. This project represents a small step forward in the journey of utilizing data to make informed decisions in campus placement scenarios.
- In conclusion, this internship in data science project has not only advanced our understanding
  of predicting campus placements but it has also demonstrated the power of data-driven
  decision-making. The skills and methodologies employed in this project can be applied to
  various domains, making data science a valuable tool in the modern world.