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**

# ANNAMALAI UNIVERSITY

## 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 Coordinator Professor & Head

**Department of CSE Department 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.

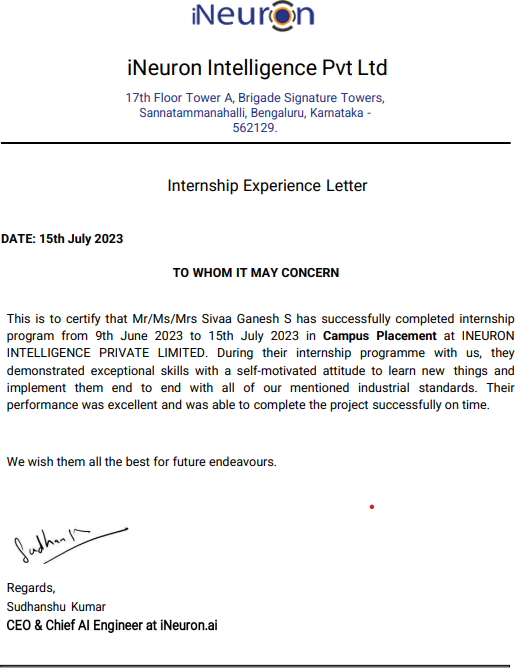
### COURSE OUTCOMES:

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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mapping of Course Outcomes with Programme Outcomes** | | | | | | | | | | | | |
|  | **P O**  **1** | **PO 2** | **PO 3** | **PO 4** | **PO 5** | **PO 6** | **PO 7** | **PO 8** | **PO 9** | **PO**  **1 0** | **PO**  **1 1** | **PO**  **1 2** |
| **CO1** | 1 | 1 | - | - | - | - | - | - | - | - | - | - |
| **CO2** | 1 | 2 | 2 | - | - | - | - | - | - | - | - | 3 |
| **CO3** | 1 | - | 2 | 1 | 2 | - | - | - | - | - | - | - |
| **CO4** | 1 | - | - | - | - | - | - | - | 2 | - | 2 | 1 |
| **CO5** | 1 | - | - | - | 2 | - | - | - | - | 3 | - | - |

# Internship Completion Certificate



## 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.
2. **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



Start

EDA on jupyter

notebook

Execption and

logger file

Data Ingestion

Data

transformation

Data

preprocessing

Feature

Engineering

Hyperparameter

Tuning

Model building

Model Evaluation

Streamlit app

Push to github

pipelining the

data preprocessing

Deployment

Exit

**Deployment Process**





Start Application



User inputs



Submit detail



Preprocessing



Stop Application



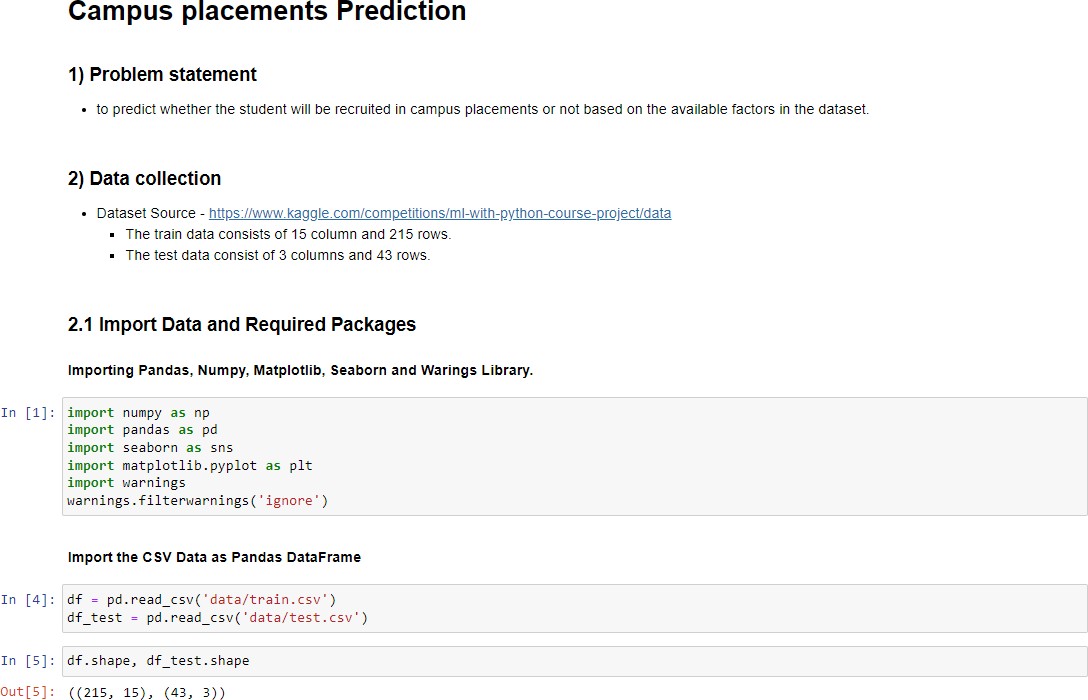
Display it in web interface

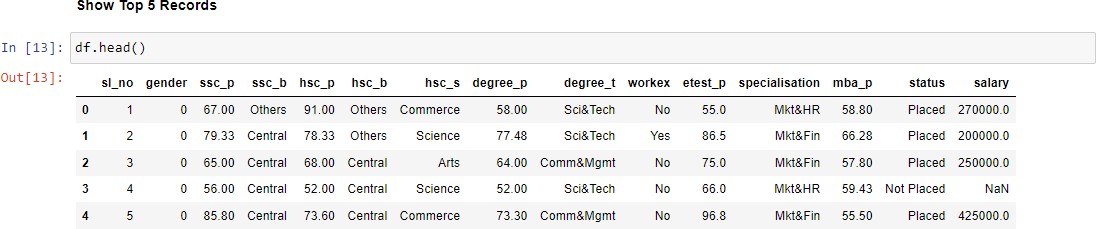


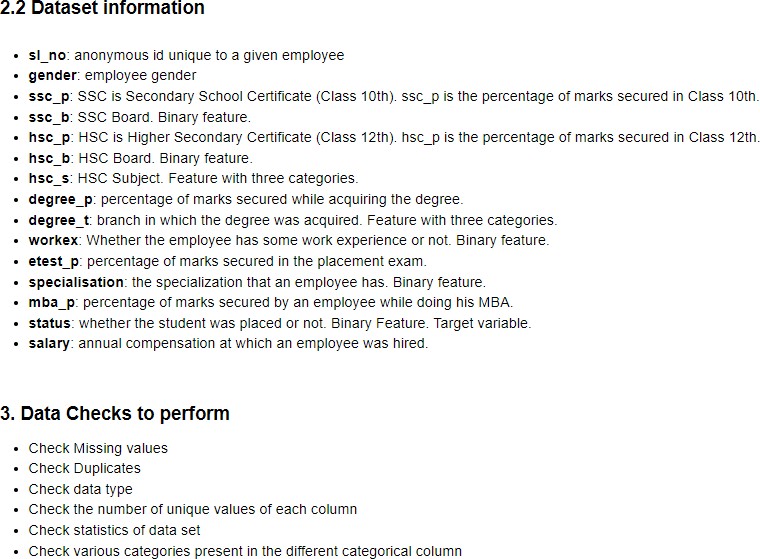
Predict the result

# Source Code

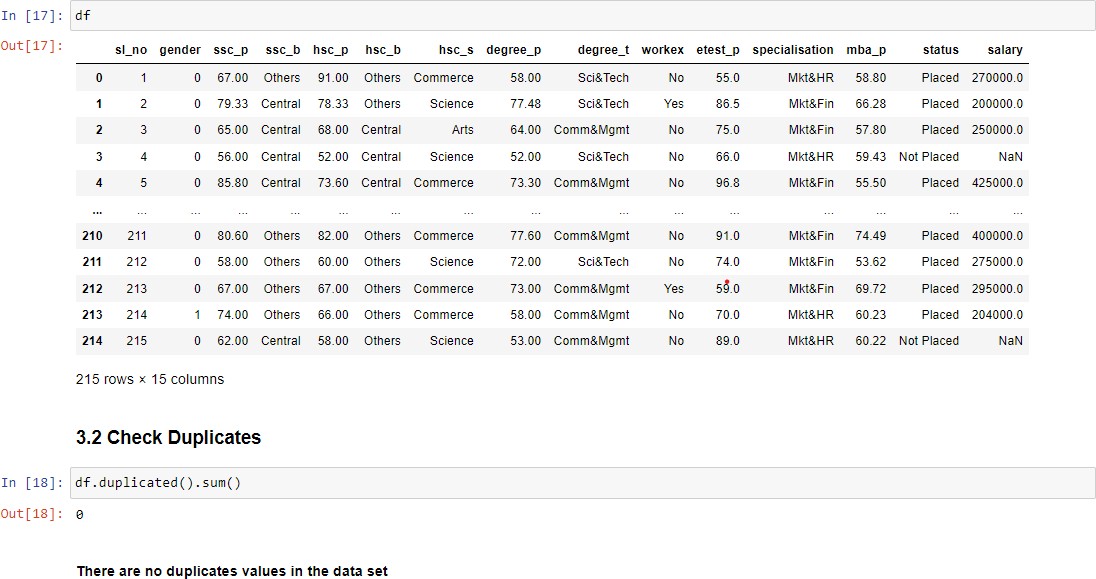
**Exploratory Data Analysis:**

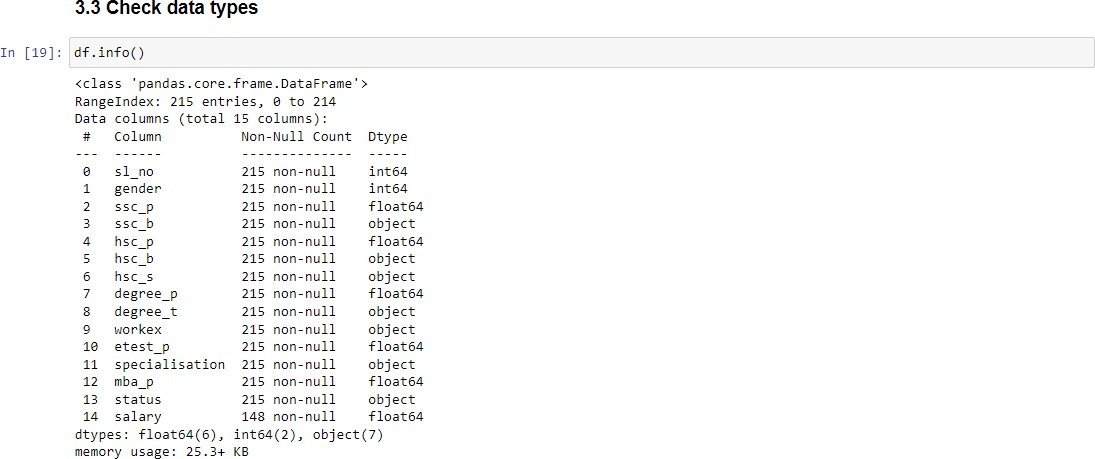


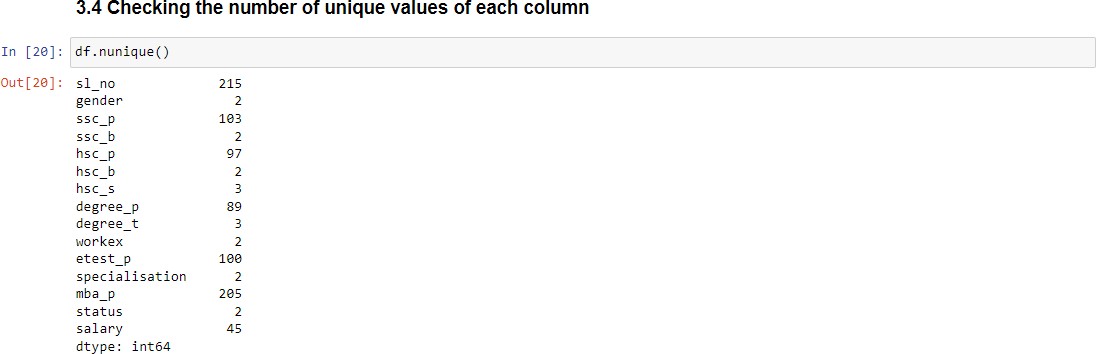


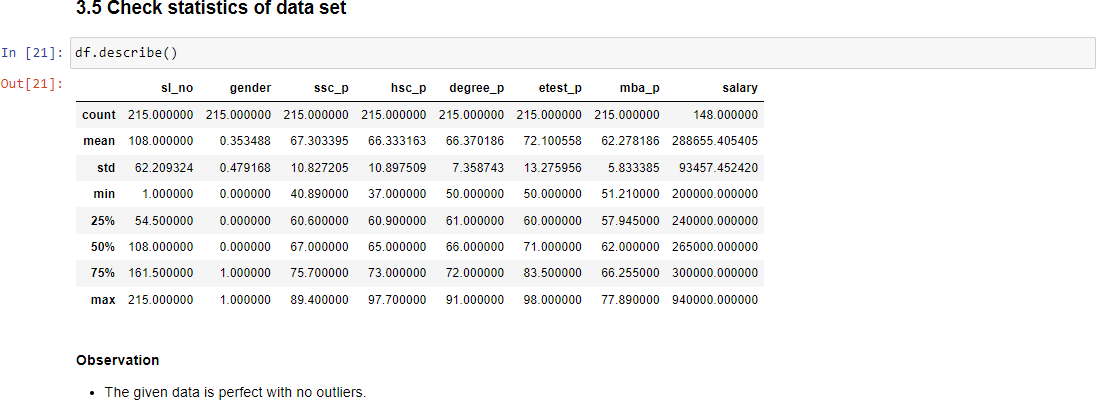


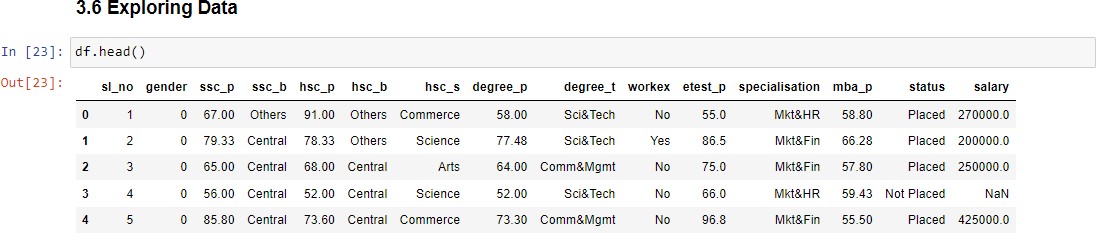


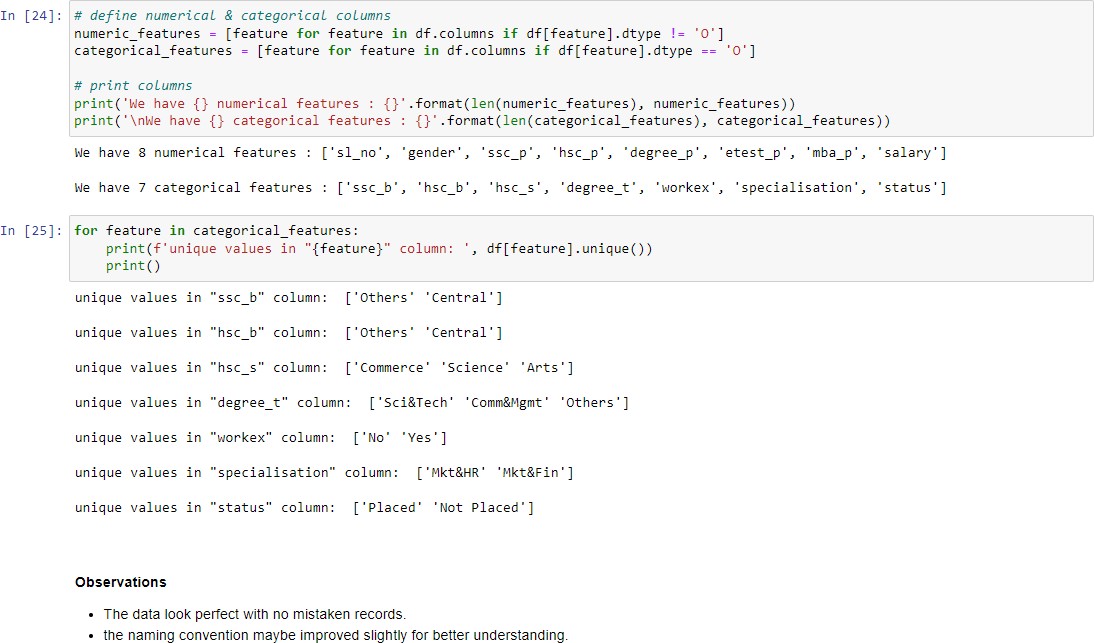




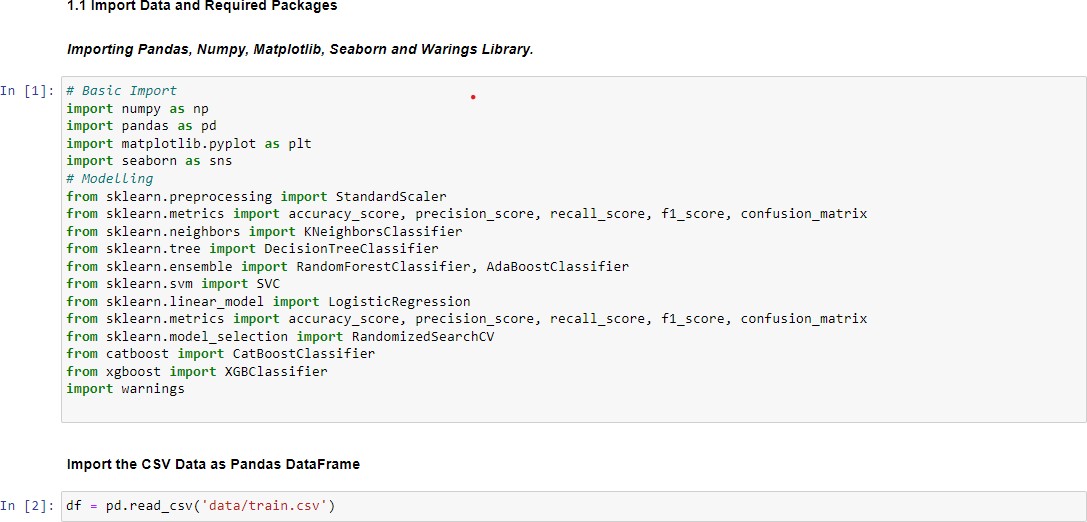


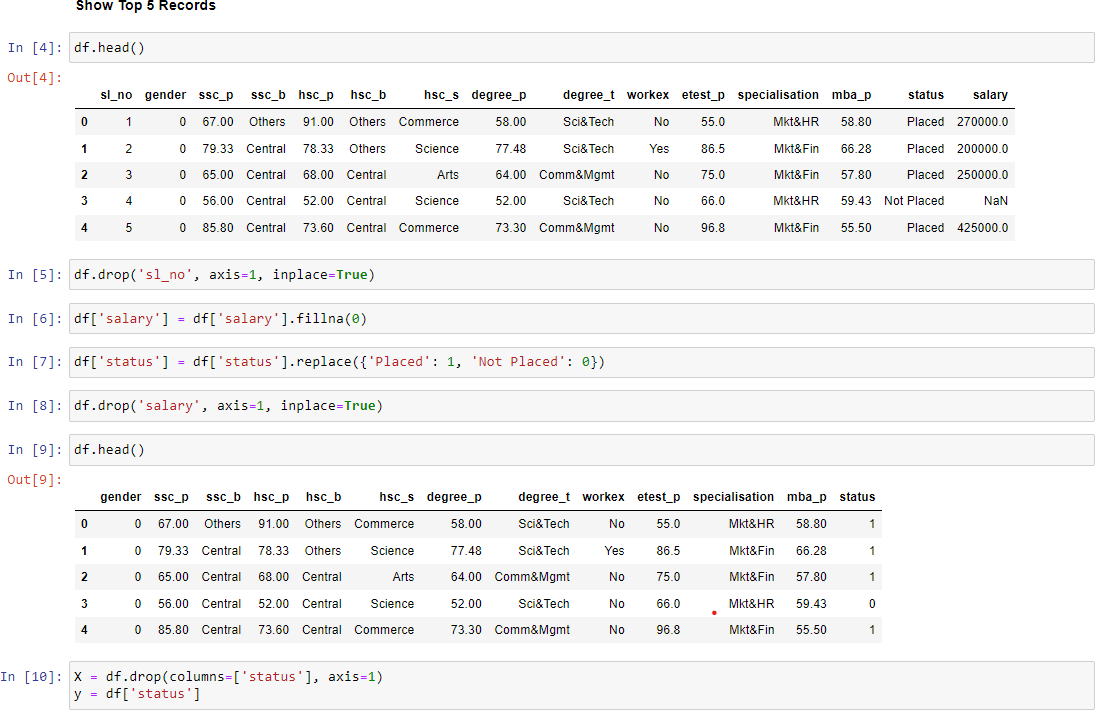




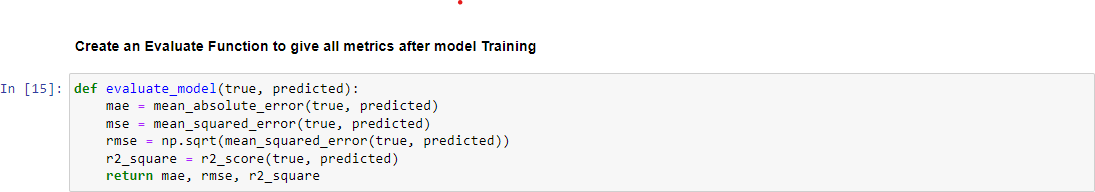


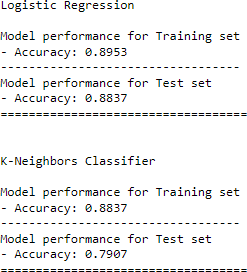
**Model Training :**

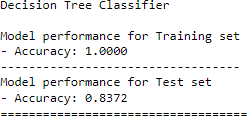


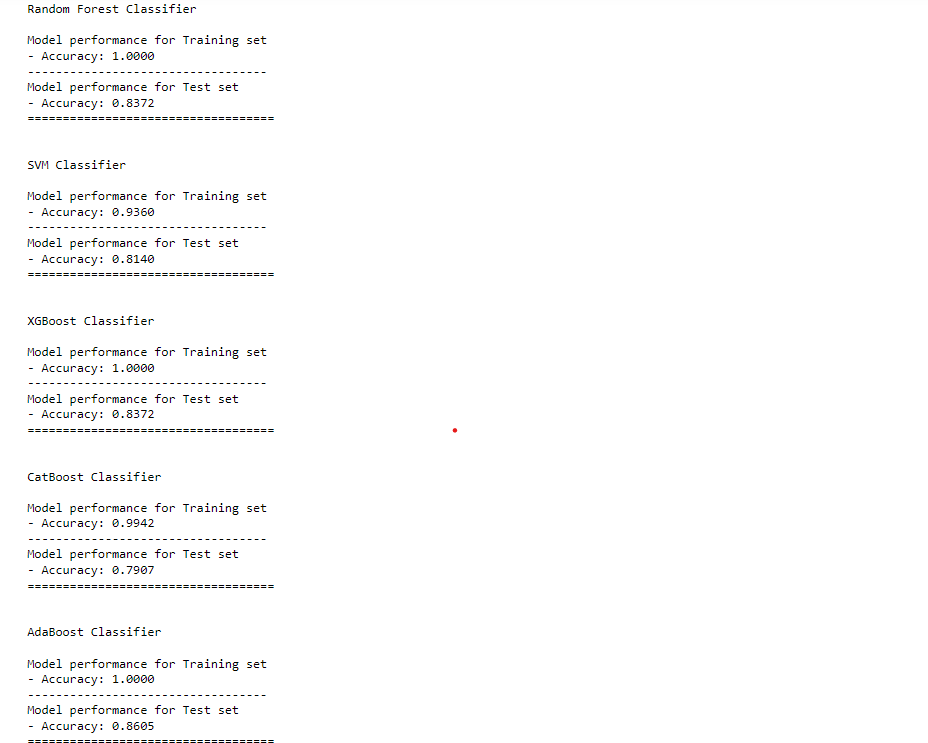


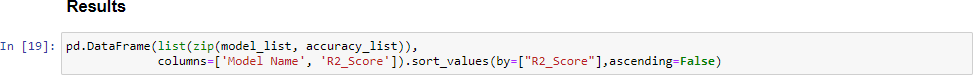
















**Data\_Ingestion.py**

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"

],axis=1)

axis=1)

input\_feature\_train\_df=train\_df.drop(columns=[target\_column\_name target\_feature\_train\_df=train\_df[target\_column\_name].values input\_feature\_test\_df=test\_df.drop(columns=[target\_column\_name], 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(

ile\_path,

file\_path=self.data\_transformation\_config.preprocessor\_obj\_f 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

)]

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](http://www.kaggle.com/competitions/ml-with-python-course-)-with-python-course[-](http://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).')

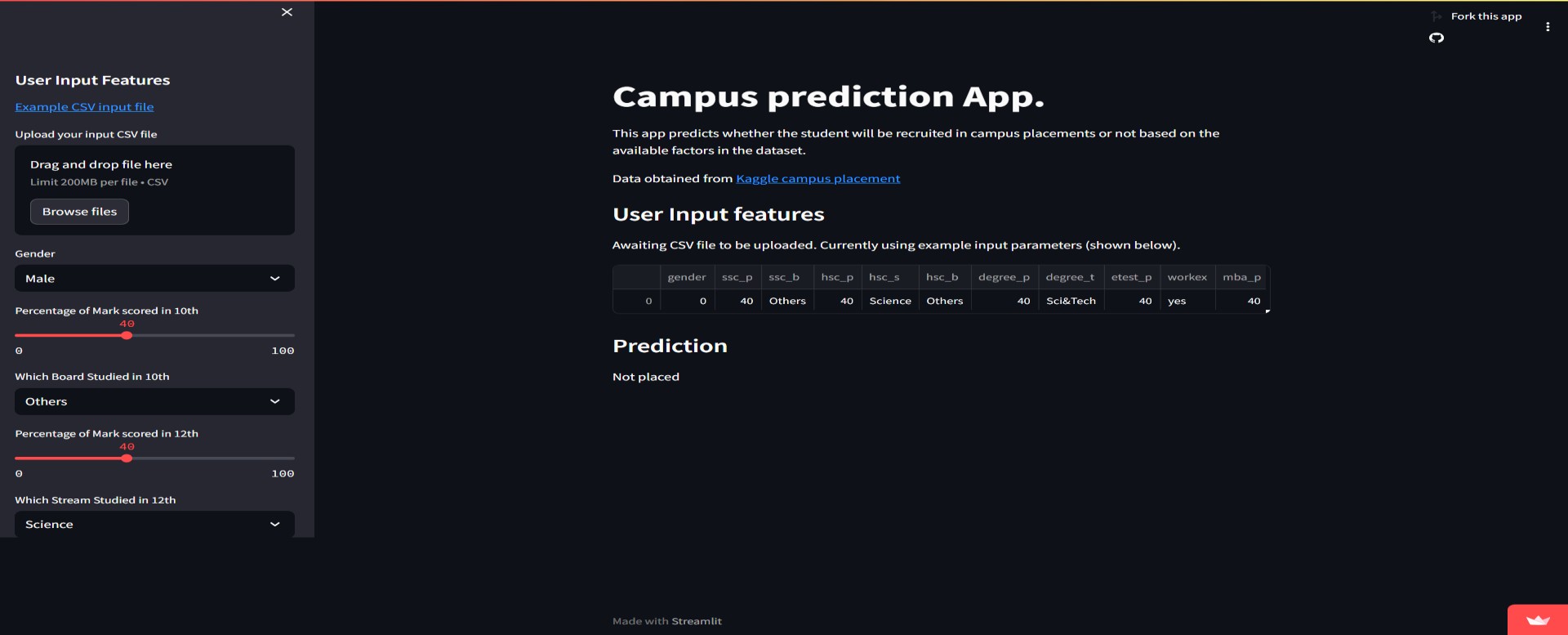
st.write(input\_df) predict\_pipeline = PredictPipeline()

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

### Programming Language -

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

### 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/>

* 1. **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 - <https://www.geeksforgeeks.org/machine-learning/>
  2. **Streamlit :** Streamlit is an open-source app framework for Machine Learning and Data Science teams. Create beautiful web apps in minutes. https://docs.streamlit.io/
  3. **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. [https://docs.github.com](https://docs.github.com/)
  4. **Referred GitHub repository :** Referred project for hyper parameter tuning and folder structure

<https://github.com/krishnaik06/mlproject>

* 1. **Referred YouTube playlist :** Referred playlist to learn about the concept of modularizing code, data pipelining and various components https://youtube.com/playlist?list=PLZoTAELRMXVPS- dOaVbAux22vzqdgoGhG&si=RAdzHEaiQTF8LeSV

# 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.