

1.INTRODUCTION

1.1 Project Overview

1.2 Introduction of ML and UI

1.3 LITERATURE SURVEY

1.4 Existing problem

1.5 Proposed Solution

Advantages and Disadvantages

2.THEORETICAL ANALYSIS

2.1 Block Diagram,
Project Structure

2.2 Data Collection and Preparation,
Activities.

3. SYSTEM SPECIFICATIONS

3.1 Software Equipment's

3.2 Minimum Hard ware equipment's

4. Result

Input and Output

Predicting the outcomes

5. CONCLUSION

INTRODUCTION

1.0 Project overview

Identifying Patterns and Trends in Campus Placement Data Using Machine Learning Campus recruitment is a strategy for sourcing, engaging and hiring young talent for internship and entry-level positions. College recruiting is typically a tactic for medium- to large-sized companies with high-volume recruiting needs, but can range from small efforts (like working with university career centers to source potential candidates) to large-scale operations (like visiting a wide array of colleges and attending recruiting events throughout the spring and fall semester). Campus recruitment often involves working with university career services centers and attending career fairs to meet in-person with college students and recent graduates. Our solution revolves around the placement season of a Business School in India. Where it has various factors on candidates getting hired such as work experience, exam percentage etc., Finally it contains the status of recruitment and remuneration details.

We will be using algorithms such as KNN, SVM and ANN. We will train and test the data with these algorithms. From this the best model is selected and saved in .pkl format. We will be doing flask integration and IBM deployment.

This paper focuses on the trend analysis of university placement. trend analysis means an aspect of technical analysis that tries to predict future movement on past data. Students are the main stakeholders of universities and their performance plays a significant role in a country's social and economic growth by producing creative graduates, innovators and entrepreneurs. a million of university and colleges provides the technical education nowadays. But few of them land in good jobs as compared to the total. A university placement plays a very important role to get more admission and maintain the reputation in the world. Authors consider some of the important attributes of students like company, package, location etc. a machine learning plays the most important

role to find out the performance, define the trend or predict the placement for the upcoming year. The dataset considered for this work is the CSE department in which we have initially considered four attributes of each student

i.e., 10th marks, 12th / Diploma marks, BTech and MTech marks. With the help of deep neural network classifier and neural networks have been known to successfully predict the future of student, out of 50 records 37 are used for training and 13 used for testing purpose and we got 77% accuracy while testing.

1.1 INTRODUCTION TO MACHINE LEARNING

Machine learning is a method of data analysis that automates analytical model building. These models help you to make a trend analysis of university placements data, to predict a placement rate for the students of an upcoming year which will help the university to analyze the performance during placements. Many students look at universities as a means of investment which can help them make a great future by getting placed in good companies and which will relieve their stress and unease from their lives before graduating from the university. The trend will also help in giving the companies reasons as to why they should visit university again and again. Some attributes play the very important role while analyzing the student for e.g., Student's name, Department, Company, Location and Annual package. So, classification can help you to classify those data and clustering helps to make the clusters department wise. In this paper we have used neural networks to predict the upcoming student placement and got 77% of accuracy while testing where iteration is 1000. Through extensive trend analysis of various complex data collected from different sources, we can demonstrate that our analysis can provide a good pragmatic solution for future placement of students. Machine Learning (ML) is that field of computer science with the help of which computer systems can provide sense to data in much the same way as human beings do. In simple words, ML is a type of artificial intelligence that extracts patterns out of raw data by using an algorithm or method. The main focus of ML is to allow computer systems learn from experience without being explicitly programmed or human intervention. Two definitions of Machine Learning are offered. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed." This is an older, informal definition.

We encounter machine learning models every day. For example, when Netflix recommends a

show to you, they used a model based on what you and other users have watched to predict what you would like. When Amazon chooses a price for an item, they use a model based on how similar items have sold in the past. When your credit card company calls you because of suspicious activity, they use a model based on your past activity to recognize anomalous behavior. In the statistical context, Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. While Machine Learning involves concepts of automation, it requires human guidance. Machine Learning involves a high level of generalization in order to get a system that performs well on yet unseen data instances. In general, any machine learning problem can be assigned to one of two broad classifications: Supervised learning and Unsupervised learning. As the data is increasing daily due to digitization in the banking sector, people want to apply for loans through the internet. Machine Learning (ML), as a typical method for information investigation, has gotten more consideration increasingly. Individuals of various businesses are utilizing ML calculations to take care of the issues dependent on their industry information. Banks are facing a significant problem in the approval of the loan. Daily there are so many applications that are challenging to manage by the bank employees, and also the chances of some mistakes are high. Most banks earn profit from the loan, but it is risky to choose deserving customers from the number of applications. There are various algorithms that have been used with varying levels of success. Logistic regression, decision tree, random forest, and neural networks have all been used and have been able to accurately predict loan defaults. Commonly used features in these studies include credit score, income, and employment history, sometimes also other features like age, occupation, and education level.

1.2 INTRODUCTION TO UI INTERFACE

HTML stands for Hypertext Markup Language. It is a standard markup language for webpage creation. It allows the creation and structure of sections, paragraphs, and links using HTML elements (the building blocks of a web page) such as tags and attributes

HTML has a lot of use cases, namely:

Web development. Developers use HTML code to design how a browser displays web page elements, such as text, hyperlinks, and media files.

Internet navigation. Users can easily navigate and insert links between related pages and websites as HTML is heavily used to embed hyperlinks.

Web documentation. HTML makes it possible to organize and format documents, similarly to Microsoft Word.

CSS (Cascading Style Sheets) is used to style and layout web pages — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features. This module provides a gentle beginning to your path towards CSS

mastery with the basics of how it works, what the syntax looks like, and how you can start using it to add styling to HTML.

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

User Interface (UI) defines the way humans interact with the information systems. In Layman 's term, User Interface (UI) is a series of pages, screens, buttons, forms and other visual elements that are used to interact with the device. Every app and every website have a user interface.

LITERATURE SURVEY

The author presents the development of placement predictor system (PPS) using logistic regression model. Based on the student scores in matriculation, senior secondary, subjects in various semesters of technical education and demographics, PS predicts the placement of a student in upcoming recruitment session. The author used logistic regression and Gradient Descent Algorithm to implement placement predictor system

- 1) The student placement is identified by using Naïve Bayesian Classification, Multilayer Perception, and C4.5 Tree. The best algorithm based on the placement data is Naïve Bayes Classification with an accuracy of 86.15%. This paper focuses on the classification of the student for the placement process in an engineering college. So author suggests a system which makes the work of prediction of student's placement easy. For this, they suggest to use ID3 algorithm- data mining classification algorithm. They have compared CHAID, C4.5 and ID3.
- 2) Using the student datasets and attributes, predictions are made using the Data Mining Algorithm "Fuzzy logic" and "K nearest neighbor (KNN)". The results obtained from each approach is then compared with respect to their "performance" and "accuracy".
- 3) This paper focuses on Predicting the performance of the M.sc student. author collect 5 Different department data and apply Naïve Bayes and 1-NN algorithms on It. They have got accuracy 85.71%. Parameter they have considered as sex, age group, marital status, bachelor, another master. They have used Weka tool to find out the accuracy.
- 4) Writer proposed the concept of the teaching of machine learning techniques using software Weka as a possible solution to integrate this issue into the study plan based on the example of problem-based learning. Techniques used linear regression Logistic regression Factor analysis Cluster analysis (hierarchical and non-hierarchical) Naive Bayes algorithm k-nearest neighbors.
- 5) Predicting the success or failure of a student in a course or program like how many students got cleared course or program in given time and the first attempt is a problem that has recently been addressed using data mining techniques k-

Nearest Neighbors, Random Forest, Classification and Regression Trees, Support Vector Machines, Naive Bayes and for regression they used Ordinary Least Squares.

- 6) Author implemented a system which was used to analyze the performance of students using k-means clustering algorithm. They cluster the student marks according to their per-centage. i.e., 75% Dist. and above, 60% first class. The parameter considered are 10th 12th B. Tech, project, internship and skill set. The algorithm used is K-means and RapidMiner studio is used.

2.1 EXISTING PROBLEM

1. Limited Staff

Most of the Campus Recruitment drives have a fixed pattern. Companies visit colleges with a small team that does all the hiring. Now since the staff is outnumbered by the high number of applicants, it poses a lot of problems in managing the whole drive. Candidate engagement is a big challenge and sometimes few candidates miss to be properly notified by the company about the further process.

All this makes the process extremely slow, inefficient, and complicated.

2. Lack of proper branding

A healthy culture in any Company involves diversity and inclusivity, something which can only be possible by having Employees who can balance their work and be an effective contributor to the company at the same time. All this starts with hiring freshers because they are raw and can be trained from scratch.

But since Companies are on a time crunch, most of them aren't able to visit most colleges or even if they do so, they look to wrap up the drive fast which invites a lot of problems. Remember that how a Company conducts Campus Drives also contributes greatly to its branding. An outdated Campus Hiring pattern can easily put your Company behind your competitors.

3. High Candidates numbers

Let's face it - in any Campus Drive, thousands of candidates will be screened. Usually, Companies visit campuses with 2-4 members. That means the staff is outnumbered while screening the candidates. Any Employee driven Company will not promote an excessive burden

on any of its employee. Still, screening 1000s of candidates in a matter of 2-3 days is a big challenge. Moreover, the screening has to be done properly so that deserving candidates aren't left behind.

Establishing proper communication with thousands of candidates all at once is another demanding task to do.

4. Negative Candidate experience

As the hiring industry is candidate-driven, the first impression during Campus Drives does a majority of the work. Any candidate will prefer a company that uses modern tools in hiring than a company that uses traditional methods and a lot of time. Moreover, if a company isn't properly conducting the Campus drives, chances are they might miss out on quality talent.

5. Importance of a Resume

One important document that flips the game for any candidate is the Resume. In any Campus Drive, it is the go-to document - both for students as well as the Recruiters. Yes, things like an impressive Resume and where a student did their internship from matters a lot, a candidate is more complex than that. Companies often fail to look beyond the resume of a candidate and make hiring decisions, hence missing out on quality talent even after personally visiting the Campus.

They should have a system that emphasizes on the overall personality of the candidate and not just one document.

2.2 PROPOSED SOLUTIONS

To deal with the problem, we developed automatic identifying patterns and trends in campus placement using machine learning techniques. We will train the machine with previous dataset. so machine can analyze and understand the process. Then machine will check for eligible applicant and give us result.

When it comes to learning technology, we should be aware of the pros and cons of that technology. The reason is so that we can understand the capabilities of that subject.

That is exactly what we are doing here. Understanding the advantages and disadvantages of Machine Learning will help us to unlock many doors.

The advantages of Machine Learning are vast. It helps us to create ways of modernizing

technology. The disadvantages of Machine Learning tell us its limits and side effects. This helps us to find different innovative ways to reduce these problems.

Advantages:

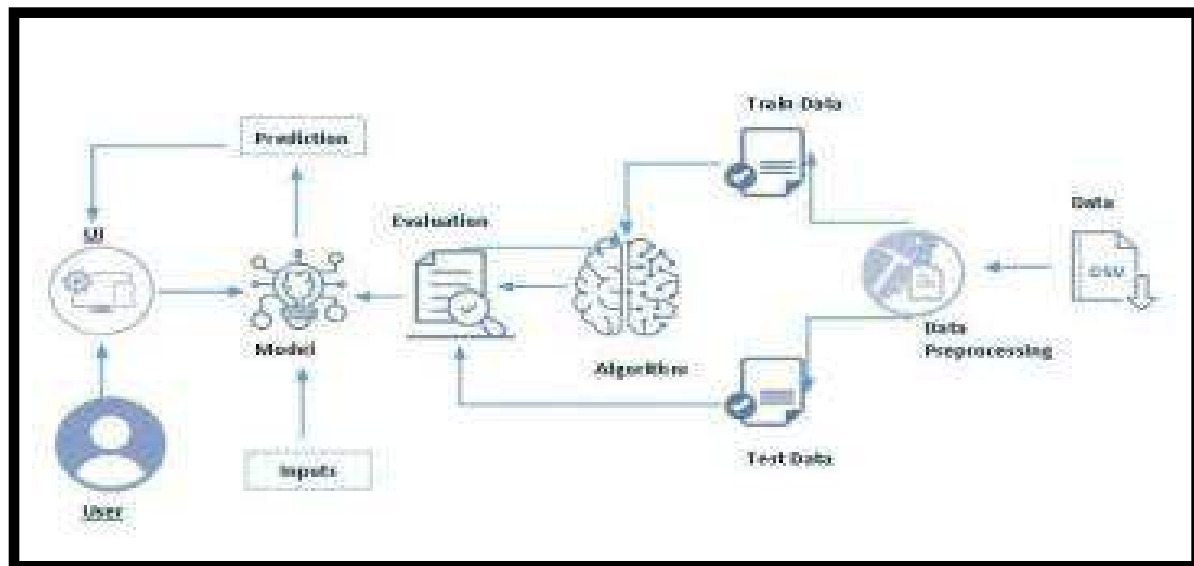
- Automation of Everything.
- Wide Range of Applications.
- Scope of Improvement.
- Efficient Handling of Data.
- Best for Education.

Disadvantages:

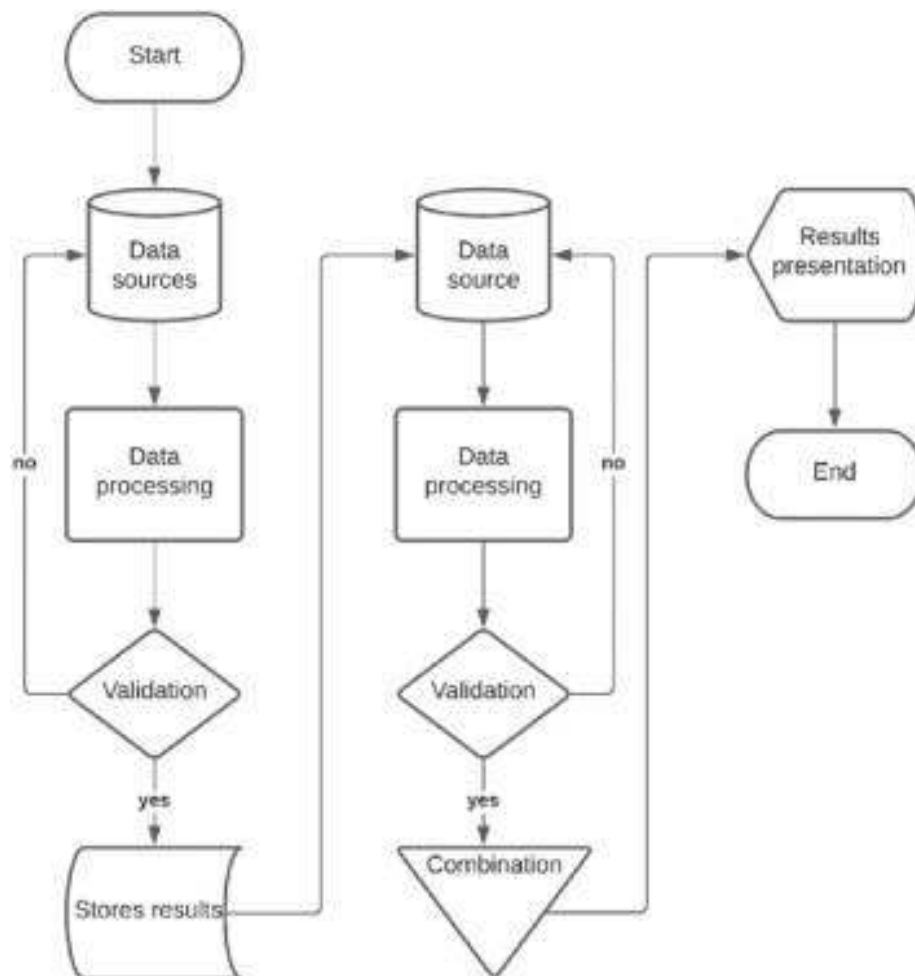
- Possibility of High Error.
- Algorithm Selection.
- Data Acquisition.
- Time and Space.

THEORETICAL ANALYSIS

3.0 BLOCK DIAGRAM:



Project structure



System architecture

The architectural design consists of the various algorithms/ Methods in machine learning. In the first phase student's data will be collected from the university. In the second phase, data will be cleaned manually by removing duplicate and unnecessary columns. In the third phase the various techniques are applied to discover the knowledge. To predict the student's placement for upcoming year, the following algorithms are used to build the prediction model. Decision tree Support vector machine Neural networks

DATA SET CONSTRUCTION:

The dataset was collected from Department of CSE of an engineering college in Pune. A dataset has 50 entries in which we have considered 37 entries for training purpose and 13 for testing. The attribute we have considered i.e., 10th marks, 12th/Diploma marks, UG and PG marks. A programming language python and open-source software library Tensor flow are playing important role to predict the student whether he/she will be placed or not in upcoming year placement.

Learning Parameters

Table 1: Parameter Used for Model
No of layers 3

No of hidden layers No of Nodes1 10

2 20

3 10.

DATA COLLECTION AND PREPARATION:

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Collect the data set:

As we have understood how the data is, let's pre-process the collected data. The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps

- Handling Missing data
- Handling Categorical data
- Handling missing data
-

Activity Activity3.1.1: Handling missing values

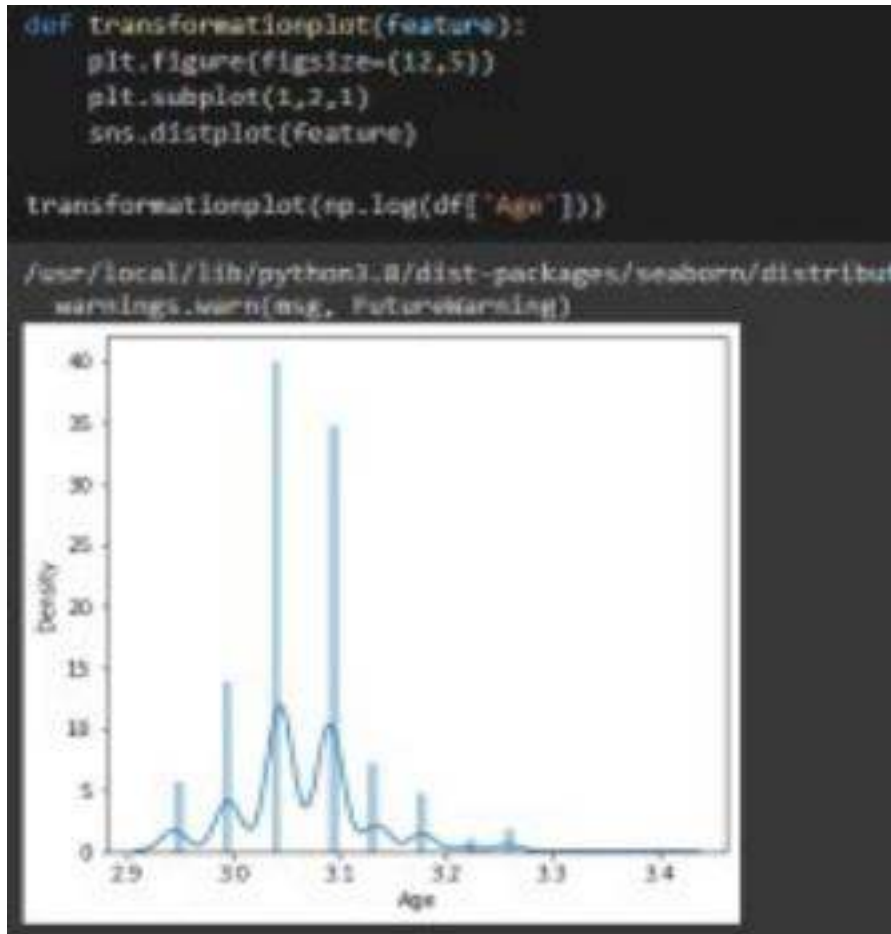
Let's find the shape of our dataset first. To find the shape of our data, the `df.shape` method is used. To find the data type, `df.info()` function is used.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2966 entries, 0 to 2965
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   Age                   2966 non-null  int64  
 1   Gender                2966 non-null  object  
 2   Stream                2966 non-null  object  
 3   Internships           2966 non-null  int64  
 4   CGPA                  2966 non-null  int64  
 5   Hostel                2966 non-null  int64  
 6   HistoryOfBacklogs     2966 non-null  int64  
 7   PlacedOrnot           2966 non-null  int64  
dtypes: int64(6), object(2)
memory usage: 185.5+ KB
```

For checking the null values, `df.isnull()` function is used. To sum those null values, we use `.sum()` function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

Activity 3.1.2: Handling outliers



Activity 3.1.3: Handling categorical values

As we can see our dataset has categorical data, we must convert the categorical data to integer encoding or binary encoding. To convert the categorical features into numerical features we use encoding techniques. There are several techniques but, in our project, we are using replacements as the distinct values are less.


```
df = df.replace(['Male'], [0])
df = df.replace(['Female'], [1])

df = df.replace(['Computer Science', 'Information Technology', 'Electronics And Communication', 'Mechanical', 'Electrical', 'Civil'],
                [0,1,2,3,4,5])
```

```
df = df.drop(['Hostel'], axis=1)

df
```

	Age	Gender	Stream	Internships	CGPA	HistoryOfBacklogs	PlacedOrNot
0	22	0	2	1	8	1	1
1	21	1	0	0	7	1	1
2	20	1	1	1	6	0	1
3	21	0	1	0	8	1	1
4	22	0	3	0	8	0	1
...
2961	23	0	1	0	7	0	0
2962	23	0	3	1	7	0	0
2963	22	0	1	1	7	0	0
2964	22	0	0	1	7	0	0
2965	23	0	5	0	8	0	1

2966 rows x 7 columns

Collect Data set :

There are many popular open sources for collecting the data. E.g.: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: <https://www.kaggle.com/code/neesham/prediction-of-placements/data>

Activity 3.1.4: Importing the libraries:

Import the necessary libraries as shown in the image. (optional) Here we have used visualizationstyle as five thirty-eight.

Let 's find the shape of our dataset first. To find the shape of our data, the df. shape method issued.To find the data type, df.info() function is used.

```
import numpy as np
import pandas as pd
import os

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import joblib
from sklearn.metrics import accuracy_score
```

Activity 3.1.5 Read the Dataset:

Our dataset format might be in .csv, excel files,. txt,. json, etc. We can read the dataset with the help of pandas. In pandas we have a function called read_csv () to read the dataset. As a parameter we have to give the directory of the csv file.

```
df = pd.read_csv(r"/content/collegePlace.csv")
df.head()
```

	Age	Gender	Stream	Internships	GPA	Hostel	HistoryOfBacklogs	PlacedOrNot
0	22	Male	Electronics And Communication	1	8	1	1	1
1	21	Female	Computer Science	0	7	1	1	1
2	22	Female	Information Technology	1	6	0	0	1
3	21	Male	Information Technology	0	8	0	1	1
4	22	Male	Mechanical	0	8	1	0	1

Exploratory Data Analysis:

Exploratory Data Analysis (EDA) is a crucial step in any data analytics project. It involves the initial analysis and visualization of the data to gain insights and understand the underlying patterns and relationships and trends in the data. It involves visualizing, summarizing, and exploring data through various statistical and graphical techniques.

The primary goal of EDA is to generate hypotheses and identify potential patterns or trends in the data, which can then be used to guide further analysis or inform decision-making. EDA can also help in identifying outliers, missing data, and other data quality issues that may need to be addressed before proceeding with further analysis.

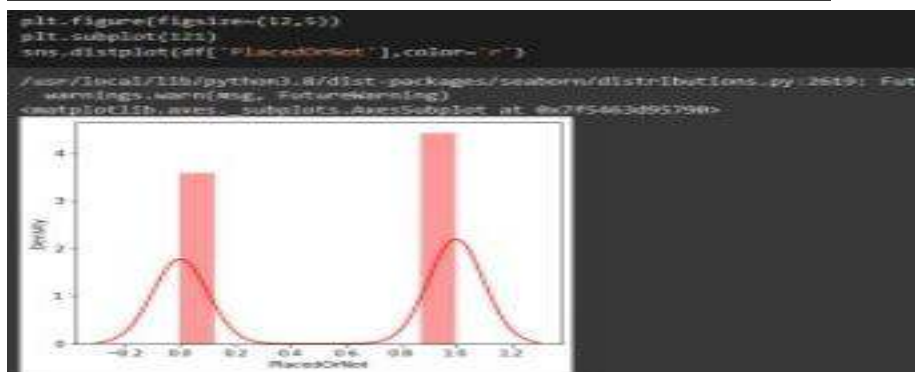
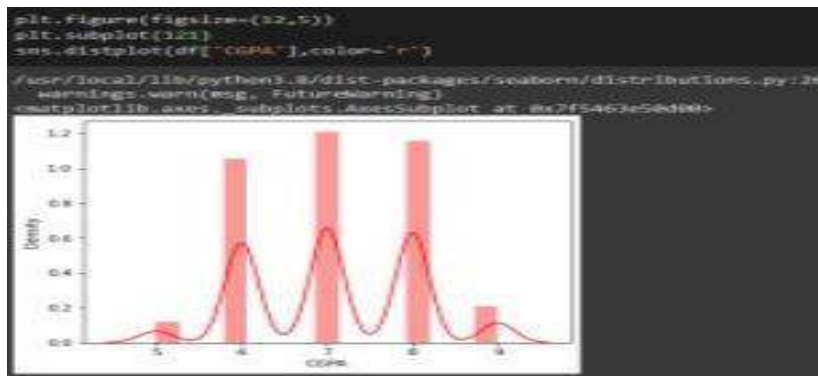
Visual Analysis:

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 3.2 1: Univariate analysis:

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and count plot.

The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplots.



In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.

From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.0

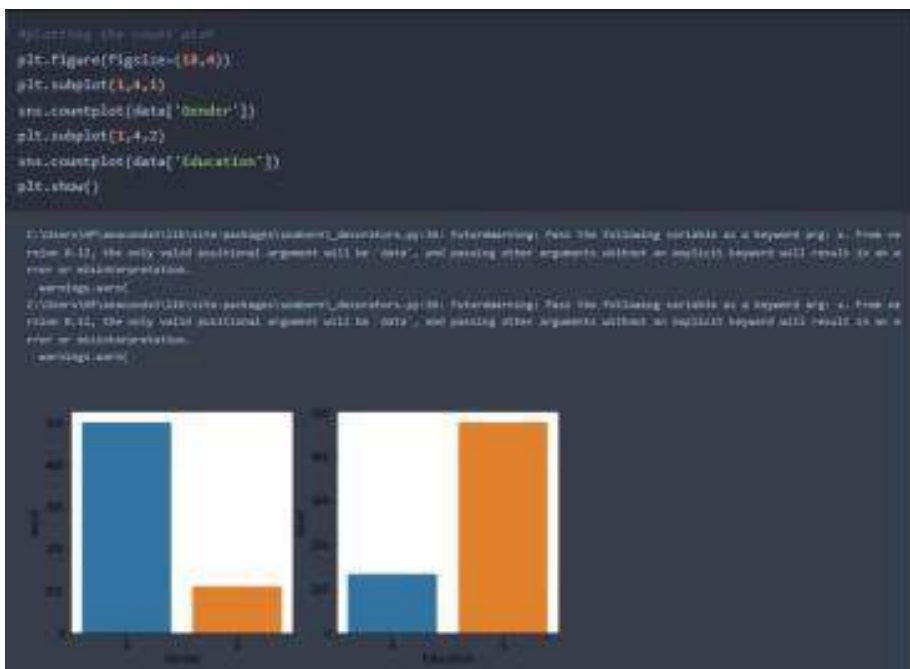
Count plot:

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot (), so you can compare counts across nested variables.

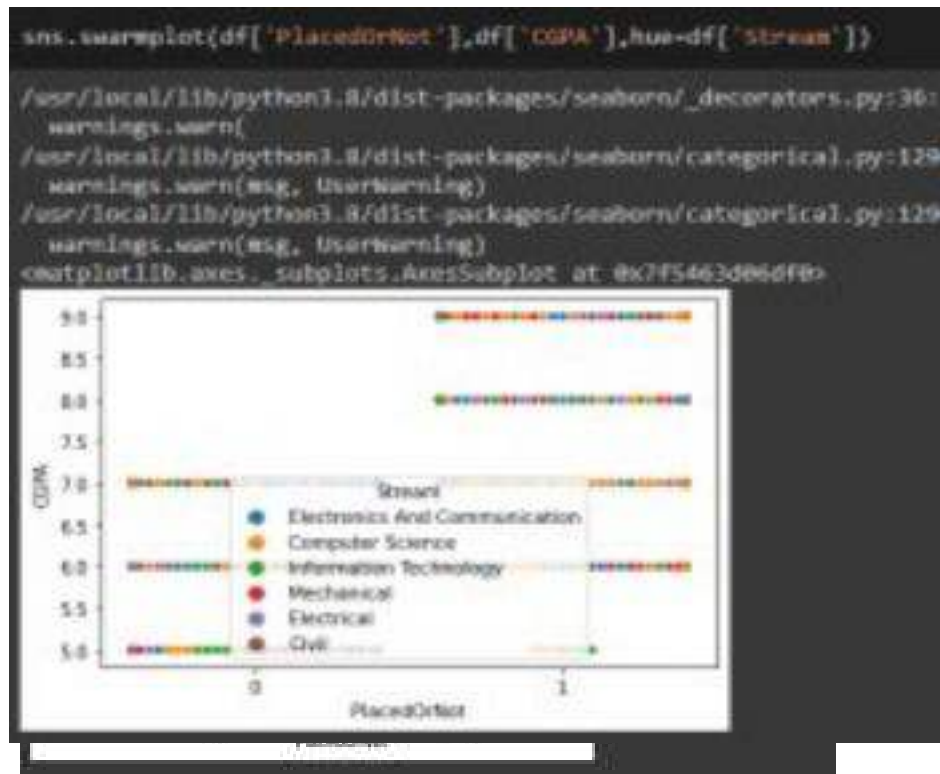
From the graph we can infer that, gender and education is a categorical variable with 2 categories, from gender column we can infer that 0-category is having more weightage than category-1, while education with 0, it means no education is an underclass when compared withcategory -1, which means educated.

Activity3. 2.2: Bivariate analysis:

Count plot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.



Activity 3.2.3: Multivariate Analysis:



In simple words, multivariate analysis is to find the relation between multiple features. Here we have used swarm plot from the seaborn package.

Scaling The Data:

Scaling is one of the important processes we have to perform on the dataset, because data measures in different ranges can lead to mislead in prediction Models such as KNN, Logistic regression needs scaled data, as they follow distance-based method and Gradient Descent concept.

```
# performing feature scaling operation using standard scaler as a part of the sklearn module  
# there different type of scaler in the sklearn  
sc=StandardScaler()  
x_bal=sc.fit_transform(x_bal)  
  
x_bal = pd.DataFrame(x_bal, columns=names)
```

We will perform scaling only on the input values. Once the dataset is scaled, it will be converted into an array and we need to convert it back to a dataframe.

We will perform scaling only on the input values. Once the dataset is scaled, it will be converted into an array and we need to convert it back to a data frame.

Splitting The Data into Train and Test:

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set. Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing. Data we are using the train_test_split () function from sklearn. As parameters, we are passing x y, test size, random state.

Model Building:

Model building is the process of developing a machine learning model to solve a specific problem or make predictions based on data. It involves selecting an appropriate algorithm, training the model on a dataset, and evaluating its performance.

Training The Model in Multiple Algorithms:

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Flow our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Activity 3.3 1: SVM model

A function named Support vector machine is created and train and test data are passed as the parameters. Inside the function, SVMClassifier algorithm is initialized and training data is passed to the model with fit () function. Test data is predicted with predict () function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```

classifier = svm.SVC(kernel='linear')

classifier.fit(X_train, Y_train)

SVC(kernel='linear')

X_train_prediction = classifier.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print('Accuracy score of the training data : ', training_data_accuracy)

Accuracy score of the training data :  0.7585497478429839

```

```

X = standardized_data
Y = df['PlacedOrNot']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, stratify=Y, random_state=2)

```

Activity 3.3.2: KNN model:

A function named KNN is created and train and test data are passed as the parameters. Inside the function, KNeighborsClassifier algorithm is initialized and training data is passed to the model with. fit () function. Test data is predicted with. predict () function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```

best_k = {'Regular':0}
best_score = {'Regular':0}
for k in range(3, 50, 2):

    ## Using Regular training set
    knn_temp = KNeighborsClassifier(n_neighbors=k)
    knn_temp.fit(X_train, Y_train)
    knn_temp_pred = knn_temp.predict(X_test)
    score = metrics.accuracy_score(Y_test, knn_temp_pred) * 100 # Get accuracy
    if score >= best_score['Regular'] and score < 100:
        best_score['Regular'] = score
        best_k['Regular'] = k
    # Instantiate the model
    # Fit the model to the training set
    # Predict on the test set
    # Store best params.

print("---Results---\nK: {} \nscore: {}".format(best_k, best_score))

## Instantiate the model
knn = KNeighborsClassifier(n_neighbors=best_k['Regular'])
## Fit the model to the training set
knn.fit(X_train, Y_train)
knn_pred = knn.predict(X_test)
testd = accuracy_score(knn_pred, Y_test)

---Results---
K: {'Regular': 7}
Score: {'Regular': 86.29128619528619}

```

Activity 3.3.3: Artificial neural network model:

Artificial neural network (ANN) is an information-processing paradigm that is inspired by way biological nervous system. Its designed as an interconnected system of processing element each with a limited numbers of input and output rather than being programmed this system learns to recognize pattern. Neural networks are constructed for a specific application, such as pattern recognition through a learning process. In biological system, learning involves adjustment to the synaptic connections between neurons. We will also be using a neural network to train the model.

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from tensorflow.keras import layers

[ ] classifier = Sequential()

#add input layer and first hidden layer
classifier.add(keras.layers.Dense(8, activation = 'relu', input_dim = 4))
classifier.add(keras.layers.Dropout(0.5))
#add 2nd hidden layer
classifier.add(keras.layers.Dense(4, activation = 'relu'))
classifier.add(keras.layers.Dropout(0.5))

#final or output layer
classifier.add(keras.layers.Dense(1, activation = 'sigmoid'))

[ ] #compiling the model

loss_l = tf.keras.losses.BinaryCrossentropy()
classifier.compile(optimizer = 'Adam', loss = loss_l, metrics = ['accuracy'])

[ ] #fitting the model
classifier.fit(x_train, y_train, batch_size = 32, epochs = 100)
```

Activity 3.3.4. Decision tree model:

A function named decision Tree is created and train and test data are passed as the parameters.

Inside the function, Decision Tree Classifier algorithm is initialized and training data is passed to the model with the. fit () function. Test data is predicted with. predict () function and saved in anew variable. For evaluating the model, a confusion matrix and classification report is done.


```
def decisionTree(x_train, x_test, y_train, y_test):
    dt=DecisionTreeClassifier()
    dt.fit(x_train,y_train)
    yPred = dt.predict(x_test)
    print('***DecisionTreeClassifier***')
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

Performance Testing & Hyperparameter Tuning:

Performance testing is the process of evaluating the performance of a system or application under specific workloads or scenarios. It involves measuring key performance metrics such as response time, throughput, and resource utilization to identify performance bottlenecks and areas for optimization.

Performance testing can be done using various techniques such as load testing, stress testing, endurance testing, and scalability testing. Load testing involves simulating user loads to test the system's ability to handle high levels of traffic. Stress testing involves testing the system beyond its expected capacity to identify its breaking point. Endurance testing involves testing the system under sustained loads to identify performance degradation over time. Scalability testing involves testing the system's ability to handle increasing levels of load by adding more resources.

Hyperparameter tuning, on the other hand, is the process of finding the optimal values for the hyperparameters of a machine learning model to improve its performance. Hyperparameters are parameters that are set before the training process, such as learning rate, batch size, and regularization, and can have a significant impact on the model's performance.

Hyperparameter tuning can be done using various techniques such as grid search, random search, and Bayesian optimization. Grid search involves testing all possible combinations of hyperparameters within a specified range. Random search involves randomly sampling hyperparameters from a specified distribution. Bayesian optimization involves using a probabilistic model to predict the performance of different hyperparameter combinations and selecting the optimal combination based on these predictions.

Overall, performance testing and hyperparameter tuning are critical processes in ensuring the optimal performance of systems and machine learning models, respectively. By identifying performance bottlenecks and finding the optimal hyperparameters, organizations can improve the user experience and achieve better results.

Performance testing and hyperparameter tuning are critical steps in building an effective machine learning model. These steps are used to optimize the performance of the model and ensure that it is accurate and reliable.

Performance testing involves evaluating the performance of the model on a separate test dataset, which has not been used in the model training process. The goal of performance testing is to determine how well the model performs on new data, and to identify any issues or areas for improvement. Common metrics used to evaluate the performance of a machine learning model include accuracy, precision, recall, F1 score, and area under the curve (AUC). Hyperparameter tuning involves fine-tuning the parameters of the machine learning algorithm to optimize its performance. Hyperparameters are parameters that are set before the model training process begins, and they can have a significant impact on the performance of the model. Examples of hyperparameters include the learning rate, regularization parameter, and number of hidden layers in a neural network.

There are several techniques for hyperparameter tuning, including grid search, random search, and Bayesian optimization. Grid search involves trying out all possible combinations of hyperparameters, while random search involves randomly sampling hyperparameters from a defined range. Bayesian optimization uses probabilistic models to estimate the performance of different hyperparameter configurations.

When conducting performance testing and hyperparameter tuning, it's important to use best practices such as cross-validation to ensure that the results are reliable and robust. It's also important to be aware of the trade-off between model complexity and model performance, and to choose the simplest model that adequately solves the problem at hand.

Testing Model with Multiple Evaluation Metrics:

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

For comparing the above four models, the compare Model function is defined.

Model Deployment:

Model deployment is the process of deploying a machine learning model into a production environment, where it can be used to make predictions on new data. It involves taking the trained model and integrating it into an application or system, and making it available to end-users.

Save the best model:

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
[ ] import pickle

pickle.dump(knn, open("placement.pkl", 'wb'))
model = pickle.load(open('placement.pkl', 'rb'))
```

Integrate With Web Framework:

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user where he has to enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages

- Building server-side script

- Run the web application

Activity 3.4.1 Building HTML pages:

For this project create one HTML file namely

1) index.html

2) index1.html

```
<section id="hero" class="d-flex flex-column justify-content-center">
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-xl-8">
        <h1>Identifying Patterns and Trends in Campus Placement Data using Machine Learning</h1>
      </div>
    </div>
  </div>
</section><!-- End Hero -->
```

3)

secondpage.html

and save it in templates folder

Activity 3.4.2: Building Html Pages(part-2)

The below code is written to fetch the details from the user using <form> tag. A submit button is provided at the end which navigates to the prediction page upon clicking.

```
<section id="about" class="about">
  <div class="container">
    <div class="section-title">
      <h2>Fill the details</h2>
    </div>
    <div class="row content">
      <div class="form">
        <form action="{% url for 'y_predict' %}" method="POST">
          <input type="number" id="sex1" name="sex1" placeholder="Age">
          <input type="number" id="sex2" name="sex2" placeholder="Gender (M/F)"/>
          <input type="number" id="sex3" name="sex3" placeholder="Stream (C/E, IT/I, ECE/2), Mech/3, ECE/4, Civil/5">
          <input type="number" id="sex4" name="sex4" placeholder="Internships">
          <input type="number" id="sex5" name="sex5" placeholder="CGPA">
          <input type="number" id="sex6" name="sex6" placeholder="Number of Backlogs">
          <input type="submit" value="Submit">
        </form>
      </div>
    </div>
  </div>
</section><!-- End About Us Section -->
```

The code for secondpage.html is as shown below. The code includes a section that provides the output on screen.

```
<section id="here" class="d-flex flex-column justify-content-center">
  <div class="container">
    <div class="row justify-content-center">
      <div class="col-xl-8">
        <div>The Prediction is : {{y}}</div>
        <div>0 represents Not-Placed </div>
        <div>1 represents Placed</div>
      </div>
    </div>
  </div>
</section><!-- End Here -->
```

Activity 3.4.3: Build python coding:

Import the libraries

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument.

```
from flask import Flask, render_template, request
app=Flask(__name__)
import pickle
import joblib
model=pickle.load(open("placement123.pkl","rb"))
ct=joblib.load('placement')
```

Activity 3.4.4: Render HTML page:

```
@app.route('/')
def hello():
    return render_template("index.html")
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

This below code retrieves the value from UI

```
app.route('/guest', methods = ['POST'])
def guest():
    sen1=request.form["sen1"]
    sen2=request.form["sen2"]
    sen3=request.form["sen3"]
    sen4=request.form["sen4"]
    sen5=request.form["sen5"]
    sen6=request.form["sen6"]

app.route('/y_predict', methods = ['POST'])
def y_predict():
    x_test = [(yo) for yo in request.form.values()]

    prediction =model.predict(x_test)

    prediction = prediction[0]

    return render_template("secondpage.html",y=prediction)
```

Here in the above code, we are routing our app to predict () function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model. Predict() function. This function returns the prediction. And this predictionvalue will be rendered to the text that we have mentioned in the submit.html page earlier.

Activity 3.4.5: Main Function:

```
app.run(debug=True)
```

- 1)Open anaconda prompt from the start menu
- 2) Navigate to the folder where your python script is.
- 3)Now type “python app.py” command
- 4)Navigate to the localhost where you can view your web page.

Click on the predict button from the top right corner, enter the inputs, click on the submit button,and see the result/prediction on the web.

```
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 346-359-021
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Copy the link from the prompt and paste it in chrome. A web page opens up as described below
Let's see how our page looks like :



We need to enter the values into the fields provided to get our prediction
Let's look how our html file looks like after entering values:

FILL THE DETAILS

22
0
2
1
9
1

Predict

Now when you click on submit button to see the predictionLet's look how our prediction looks like



SYSTEM SPECIFICATION

4.1. SOFTWARE REQUIREMENTS:

Language Tools used: MS-EXCEL

Anaconda Navigator

Jupyter Notebook

Operating System: Windows or Linux

Language: python, html

Python ide: python 2.7x and above

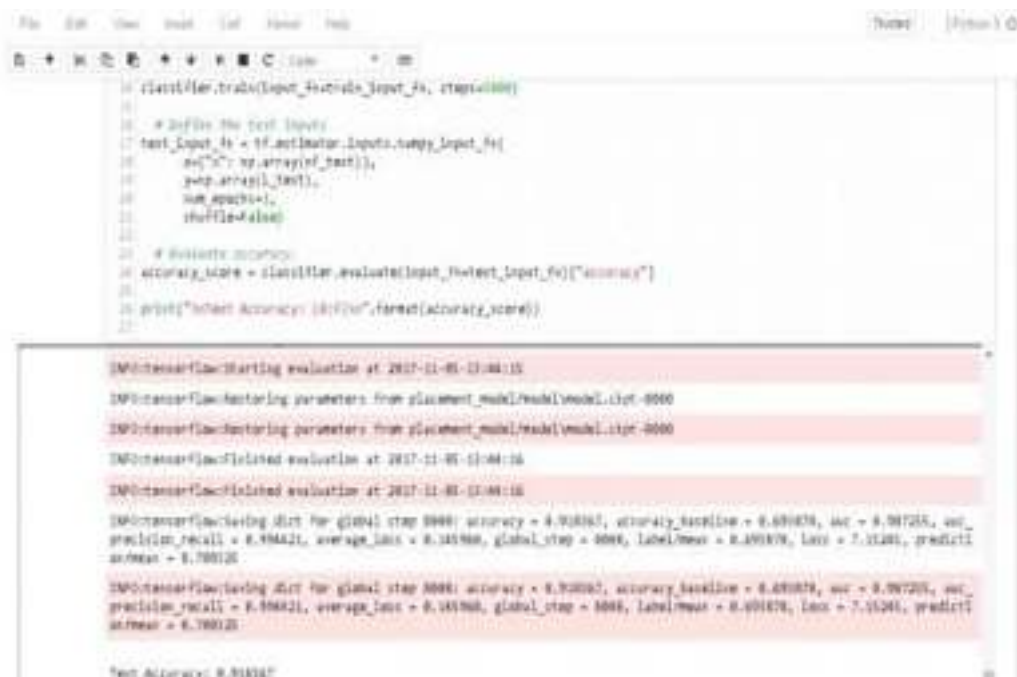
4.2. MINIMUM HARDWARE REQUIREMENTS:

Hard Disk: 500GB minimum

PROCESSOR: Intel i3 and above

Ram: 4 GB and Higher.

Result:



```
10 classifier.train(input_features,input_fe, steps=1000)
11
12 # Define the test inputs
13 test_input_fe = tf.nn.concatenate([
14     np.array([1,0,0]),
15     np.array([1,0,0]),
16     np.array([1,0,0]),
17     np.array([1,0,0])
18 ],axis=1,
19 shuffle=False)
20
21 # Evaluate accuracy
22 accuracy_score = classifier.evaluate(input_fe,test_input_fe)["accuracy"]
23
24 print("Test Accuracy: {}".format(accuracy_score))
25
```

INFO:tensorflow:Starting evaluation at 2017-11-01 13:04:15

INFO:tensorflow:Restoring parameters from placement_model/model.ckpt-0000

INFO:tensorflow:Restoring parameters from placement_model/model.ckpt-0000

INFO:tensorflow:Finished evaluation at 2017-11-01 13:04:16

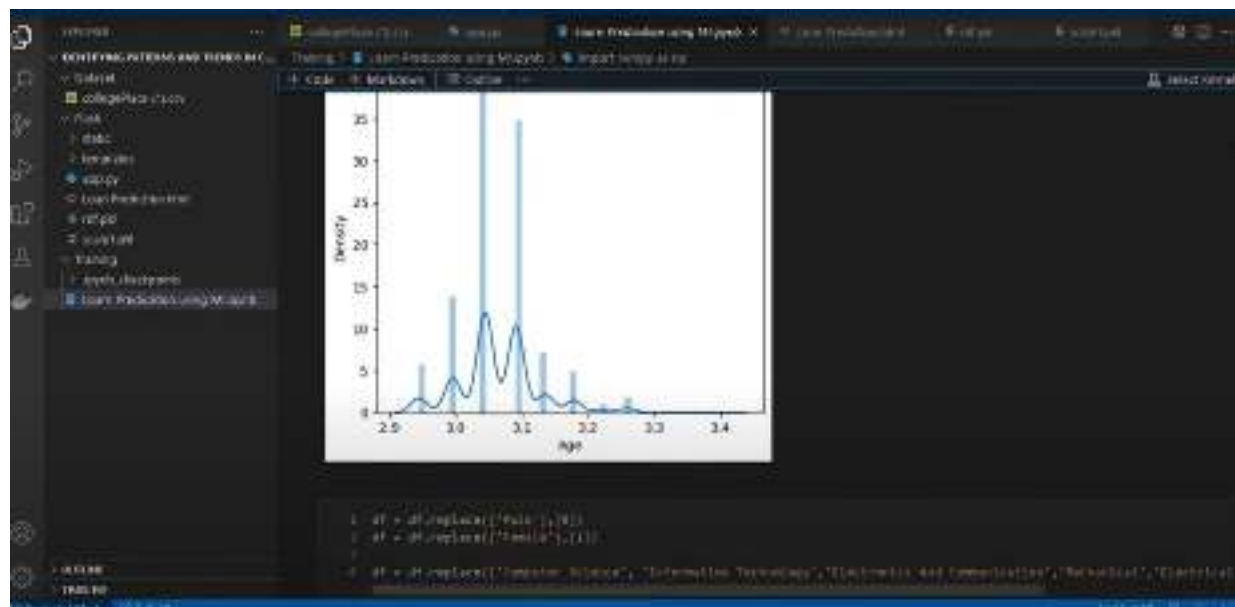
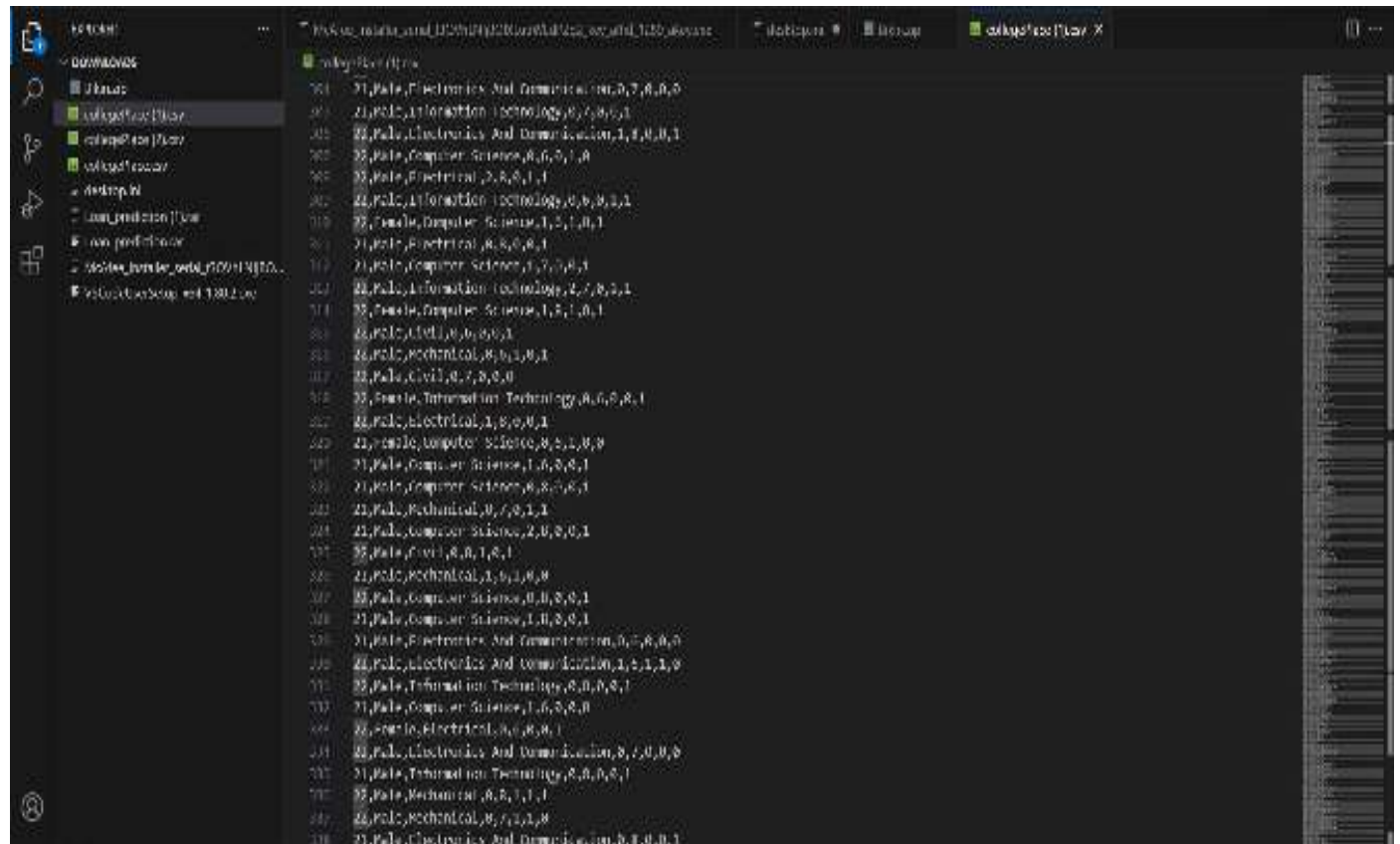
INFO:tensorflow:Finished evaluation at 2017-11-01 13:04:16

INFO:tensorflow:Saving dict for global step 0000: accuracy = 0.90861, accuracy_baseline = 0.69478, auc = 0.90725, auc_precision_recall = 0.90421, average_loss = 0.34566, global_step = 0000, label_mean = 0.49979, loss = 7.51201, predict1_mean = 0.70012

INFO:tensorflow:Saving dict for global step 0000: accuracy = 0.90861, accuracy_baseline = 0.69478, auc = 0.90725, auc_precision_recall = 0.90421, average_loss = 0.34566, global_step = 0000, label_mean = 0.49979, loss = 7.51201, predict1_mean = 0.70012

Test Accuracy: 0.90861

The above figure accuracy shows the training of data where number of iteration are 1000. We got 71% accuracy.



```

File Edit View Insert Cell Format Help
Python 3.6

1: classifier.train(input_normal_input_4e, steps=2000)
2:
3: # Define the test inputs
4: test_input_4e = tf.nn.conv2d(input_normal_input_4e,
5:                               kernel=[3, 3, 1, 1],
6:                               strides=[1, 1, 1, 1],
7:                               padding='VALID',
8:                               data_format='NHWC')
9:
10: # Evaluate accuracy
11: accuracy_score = classifier.evaluate(test_input_4e, test_input_4e, ["accuracy"])
12:
13: print("Test Accuracy: {0:f}%".format(accuracy_score))
14:
15:
16: INFO:tensorflow:Starting evaluation at 2017-11-05 12:41:16
17: INFO:tensorflow:Restoring parameters from placement_model\model.ckpt-1000
18: INFO:tensorflow:Restoring parameters from placement_model\model.ckpt-1000
19: INFO:tensorflow:Finished evaluation at 2017-11-05 12:41:16
20: INFO:tensorflow:Finished evaluation at 2017-11-05 12:41:16
21: INFO:tensorflow:Saving ckpt for global step 1000: accuracy = 0.77051, accuracy_baseline = 0.681878, auc = 0.872168, auc_j
22: rection_recall = 0.80217, average_loss = 0.704479, global_step = 1000, label/mean = 0.690878, loss = 0.8175, prediction
23: chance = 0.733961
24: INFO:tensorflow:Saving ckpt for global step 1000: accuracy = 0.77051, accuracy_baseline = 0.681878, auc = 0.872168, auc_j
25: rection_recall = 0.80217, average_loss = 0.704479, global_step = 1000, label/mean = 0.690878, loss = 0.8175, prediction
26: chance = 0.733961

```

The above figure shows the accuracy of 93% and number of iterations are 5000.

```

File Edit View Insert Cell Format Help
Python 3.6

1: shiftLeftValue
2:
3: # Evaluate accuracy
4: accuracy_score = classifier.evaluate(input_4e, test_input_4e, ["accuracy"])
5:
6: print("Test Accuracy: {0:f}%".format(accuracy_score))
7:
8:
9: INFO:tensorflow:Saving checkpoints for 1001 into model\iris_model\model.ckpt.
10: INFO:tensorflow:Step = 1000, loss = 71.9204
11: INFO:tensorflow:Step = 1000, loss = 71.9204
12: INFO:tensorflow:global_step/sec: 561.701
13: INFO:tensorflow:global_step/sec: 561.701
14: INFO:tensorflow:Step = 1100, loss = 47.7908 (0.09 sec)
15: INFO:tensorflow:Step = 1100, loss = 47.7908 (0.09 sec)
16: INFO:tensorflow:global_step/sec: 576.288
17: INFO:tensorflow:global_step/sec: 576.288
18: INFO:tensorflow:Step = 1200, loss = 48.8161 (0.07 sec)
19: INFO:tensorflow:Step = 1200, loss = 48.8161 (0.07 sec)
20: INFO:tensorflow:global_step/sec: 589.731

```

The above fig shows the 77% accuracy where amount of iteration in 2000.

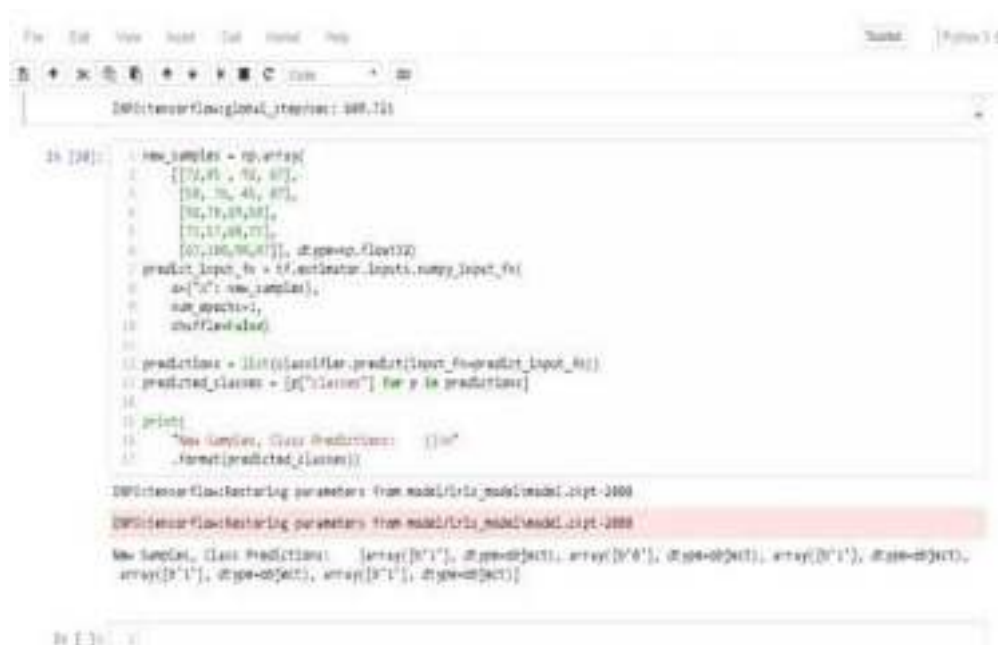
Accuracy:

```
INFO:tensorflow:Saving dict for global step 2000: accuracy = 0.734694, average_loss = 0.469289, global_step = 2000, loss = 22.9952
```

```
INFO:tensorflow:Saving dict for global step 2000: accuracy = 0.734694, average_loss = 0.469289, global_step = 2000, loss = 22.9952
```

Test Accuracy: 0.734694

Testing:



```
File Edit View Insert Cell Help
In [20]: new_samples = np.array(
1: [[71, 95, 94, 87],
2: [38, 36, 45, 87],
3: [92, 78, 87, 88],
4: [71, 87, 88, 87],
5: [67, 100, 96, 87]], dtype=np.float32)
6: predict_input_fn = tf.estimator.inputs.numpy_input_fn(
7: xs=[x]: new_samples),
8: num_epochs=1,
9: shuffle=False)
10:
11: predictions = list(classifier.predict(predict_input_fn))
12: predicted_classes = [p["classes"] for p in predictions]
13:
14: print(
15: "New Samples, Class Predictions:  {}".format(predicted_classes))
16:
17: INFO:tensorflow:Restoring parameters from model/1/1a_model/model.ckpt-2000
18: INFO:tensorflow:Restoring parameters from model/1/1a_model/model.ckpt-2000
19:
20: New Samples, Class Predictions:  [array([0]), dtype=object), array([0]), dtype=object), array([0]), dtype=object),
21: array([1]), dtype=object), array([1]), dtype=object)]
```

The above fig shows the testing prediction of trend analysis of university placement where we got the output by using DNN classifier that a new student will get placed or not. here 0 indicates he/she will not get placed and 1 indicates get placed.

The iteration sometimes called as steps in Deep neural network classifier from artificial neural networks decides the how steps we have to perform on data. Minimum steps will give you less accuracy and highest steps high accuracy but sometimes high number of steps with high accuracy leads to wrong output. That's why here we used minimum 1000 and max 5000 steps. The number of steps depends upon the size of your dataset.

PREDICTING THE OUTCOMES:

Using decision tree algorithm, the outcomes of all applicant can be stored in any file. Algorithm:

1. Import all the required python modules
2. Import the database for both TESTING and TRAINING.
3. Check any NULLVALUES are exists
4. If NULLVALUES exists, fill the table with corresponding coding
5. Exploratory Data Analysis for all ATTRIBUTES from the table
6. Plot all graphs using MATPLOTLIB module
7. Build the DECISIONTREE MODEL for the coding
8. Send that output to CSV FILE.

Future work:

A machine learning algorithm are playing a very important role while predicting something, in future student performance will tell the institute and student whether he/she will clear the subject or not.

Conclusion:

The algorithms of machine learning we have discussed are can used to find the trend of placement, which will be helpful for university to get more admission in future. We compared the algo rhythm and find out the accuracy by considering some of attributes of students. Here we used deep neural network classifier with the 1000, 2000, 5000 iterations with 71%, 77%, and 91% of accuracy.

