B.M.S COLLEGE OF ENGINEERING

(An Autonomous College under VTU, Belagavi)

Bull Temple Road, Bangalore - 560 019

A Project Report-2021-22

On

"ENGINEERING PLACEMENT PREDICTION"

Submitted as a part of Alternate Assessment for the Elective course

MACHINE LEARNING

Offered by

ELECTRONICS AND COMMUNICATIONS ENGINEERING

In association with

NOKIA NETWORKS, BANGALORE

Submitted By

NAME:	USN
1. Nandeesh	1BM19EC087
2. Niteeshkumar H V	1BM19EC098
3. Nitesh R	1BM19EC099
4. Padmaj U Naik	1BM19EC101
5. Pavan Kumar M	1BM19EC102
6. Nishit Kumar	1BM19EE077

FIC: **Dr.Suma M N** — Designation: Professor

Dr. Geetishree Mishra — Assistant Professor

TABLE OF CONTENTS

Abstract

Placement of scholars is one in every of the vital activities in academic establishments. Admission and name of establishments primarily depends on placements. Hence all institutions strive to Strengthen placement department.

Problem statement: A placement predictor is to be designed to calculate the possibility of a student being placed in a company, subject to the criterion of the company. The placement predictor takes many parameters which can be used to assess the skill level of the student. While some parameters are taken from the university level. Combining these data points, the predictor is to accurately predict if the student will or will not be placed in a company. Data from past students are used for training the predictor.

Problem solution: But the problem was to find a suitable classification algorithm that could do the job with maximum accuracy for our data set. Different algorithms have different accuracy depending on the type of problem it has to solve and the data set it has to work with. So, we decided to select four algorithms, namely Naïve Bayes, KNN, SVC, Logistic Regression, Random Forest Classifier, AdaBoost Classifier, Gradient Boosting Classifier and to compare the accuracy levels of each of these algorithms, with respect to our problem and data set. The result of this test would help us in determining which algorithm to use while implementing our predictor in the placement management system.

For this, we trained each of the algorithms with the data set that we acquired and tested it against some test data to find the accuracy of the algorithms. For each algorithm, we can easily obtain the True Positive, True Negative, False Positive and False Negative. With these four values, it was a matter of finding the accuracy using the accuracy equation.

CHAPTER 1: INTRODUCTION

Students studying in the final year of an engineering college start feeling the pressure of placement. They feel the need to know where they stand and how they can improve their skills to chances of getting a job. The main objective to build the system can be helpful to increase progress in student performance. So, students can analyze where they need to improve to secure a good placement in the near future. Placement holds the most important part of a student's life.

The prediction of placement status that students are most likely to achieve will help students to put in more hard work to make appropriate progress in stepping into a career in various technical fields. It will also help the teachers as well as placement cell in an institution to provide proper care towards the improvement of students in the duration of course. A high placement rate is a key entity in building the reputation of an educational institution. Hence such a system has a significant place in the educational system of any higher learning institution.

We aim to develop a placement predictor as a part of making a placement management system at college level which predicts the probability of students getting placed and helps in uplifting their skills before the recruitment process starts. We are using machine learning for the placement prediction. We consider Naïve Bayes, K-nearest neighbor (KNN), Support Vector Machine (SVM), Logistic Regression, Random Forest, AdaBoost and Gradient Boosting Classifier to classify students into appropriate clusters and the result would help them in improving their profile. And accuracy of respected algorithms is noted and with the comparison of various machine learning techniques, this would help both recruiters as well as students during placements and related activities.

CHAPTER 2: RELATED WORK

Senthil Kumar Thangavel, Divya Bharathi P and Abhijith Shankar [1] conducted a study to predict student placement status using two attributes, areas and CGPA results. They made use of Decision Tree Learning, SCI-Kit leaning in machine Learning here they use only two parameters such as CGPA and arrears used algorithm takes more time for prediction not efficient.

Wilton W.T. FOK, Y.S. He, H.H Au Yeung and K.Y. Law [2] Conducted a study to predict suitable course for the students, based on their behavior using Neural Network Technique. TensorFlow engine includes number of intermediate node and number of deep learning layers are adjusted and compared.

Machine Learning deals with the development, analysis and study of algorithms that can automatically detect patterns from data and use it to predict future data or perform decision making [3]. Machine learning does its functionality by creating models out of it [4]. Machine Learning has become widespread and has its applications in the field of bioinformatics, computer vision, robot locomotion, computational finance, search engine etc.

CHAPTER 3: METHODOLOGY

Out[3]:

	Age	Gender	Stream	Internships	CGPA	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

Fig 3.1: Sample Dataset

❖ **Various Machine learning models used:**

Naive Bayes

This is an easy technique for building classifiers: models that assign class labels to downside instances, painted as vectors of feature values, where class labels are drawn from a few finite sets. There is no single algorithm for training such classifiers, however a family of algorithms which is based on a standard principle: all Naive Bayes classifiers assume that value of a particular feature is independent of the value of other feature, given class variable. For example, a fruit could also be thought of be an apple if it is red, round, and about 11 cm in diameter. A Naive Bayes classifier considers every of these options to contribute severally to the likelihood that this fruit is AN apple, no matter any potential correlations between the colour, roundness, and diameter features.

K – Nearest Neighbors Classifier (KNN)

It is a simple, lazy and nonparametric classifier. KNN is favoured when all the features are continuous. KNN is additionally referred to as case-based reasoning and has been utilized in several applications like pattern recognition, statistical estimation. Classification is obtained by identifying the nearest neighbors to determine the class of an unknown sample. KNN is picked over other classification algorithms because of its high convergence speed and ease. Figure below shows nearest neighbors classification. KNN classification has two stages.

Support Vector Machine (SVM)

It is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Logistic regression

It estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas:

Logit (pi) = 1/(1+ exp(-pi))

ln(pi/(1-pi)) = Beta_0 + Beta_1*X_1 + … + B_k*K_k

Random Forest

The Random Forest or Random Decision Forest is a supervised Machine learning algorithm used for classification, regression, and other tasks using decision trees. The Random Forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly

selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

AdaBoost Classifier

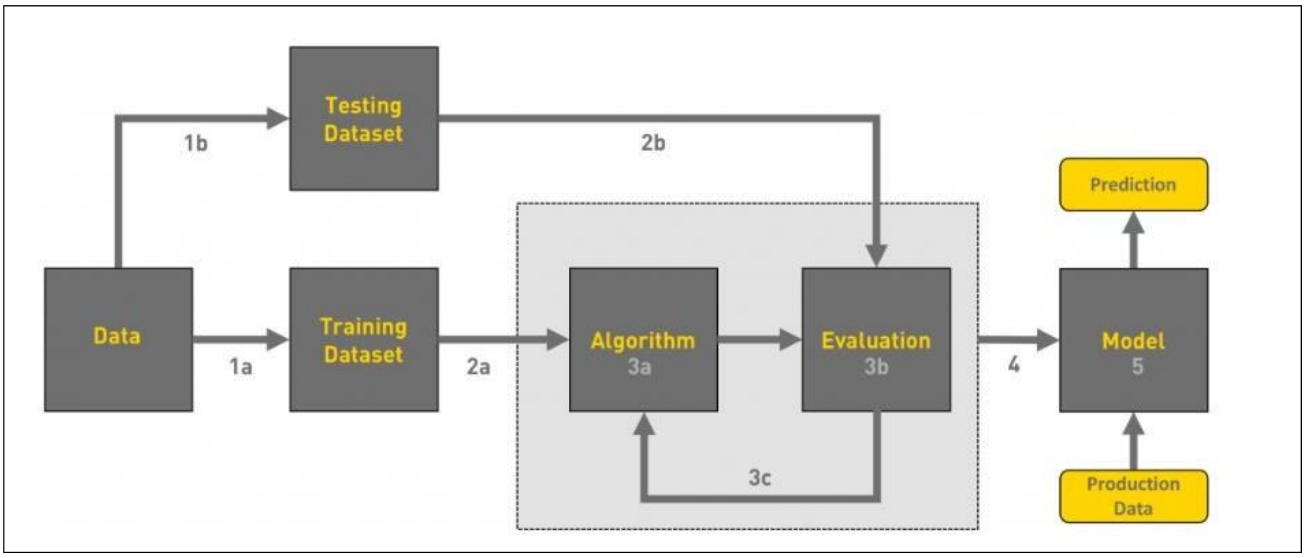
Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

AdaBoost was the first really successful boosting algorithm developed for the purpose of binary classification. *AdaBoost* is short for *Adaptive Boosting* and is a very popular boosting technique that combines multiple "weak classifiers" into a single "strong classifier". It was formulated by Yoav Freund and Robert Schapire. They also won the 2003 Gödel Prize for their work.

Gradient Descent Classifier

Gradient descent is an optimization algorithm which is commonly-used to train machine learning models and neural networks. Training data helps these models learn over time, and the cost function within gradient descent specifically acts as a barometer, gauging its accuracy with each iteration of parameter updates

CHAPTER 4: IMPLEMENTATION



4.1. Gathering Data

The process of gathering data depends on the type of project we desire to make, if we want to make an ML project that uses real-time data, then we can build an IoT system that using different sensors data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Therefore, to solve this problem Data Preparation is done.

4.2. Data pre-processing

Data pre-processing is one of the most important steps in machine learning. It is the most important step that helps in building machine learning models more accurately. In machine learning, there is an 80/20 rule. Every data scientist should spend 80% time for data pre-processing and 20% time to actually perform the analysis.

Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered

from different sources it is collected in a raw format and this data isn't feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.

Three Types of Data

1. Numeric e.g. internship, CGPA
2. Categorical e.g. gender, College stream
3. Ordinal e.g. low/medium/high

Some of the basic pre-processing techniques that can be used to convert raw data.

1. **Conversion of data:** As we know that Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features.

2. **Ignoring the missing values:** Whenever we encounter missing data in the data set then we can remove the row or column of data depending on our need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset.

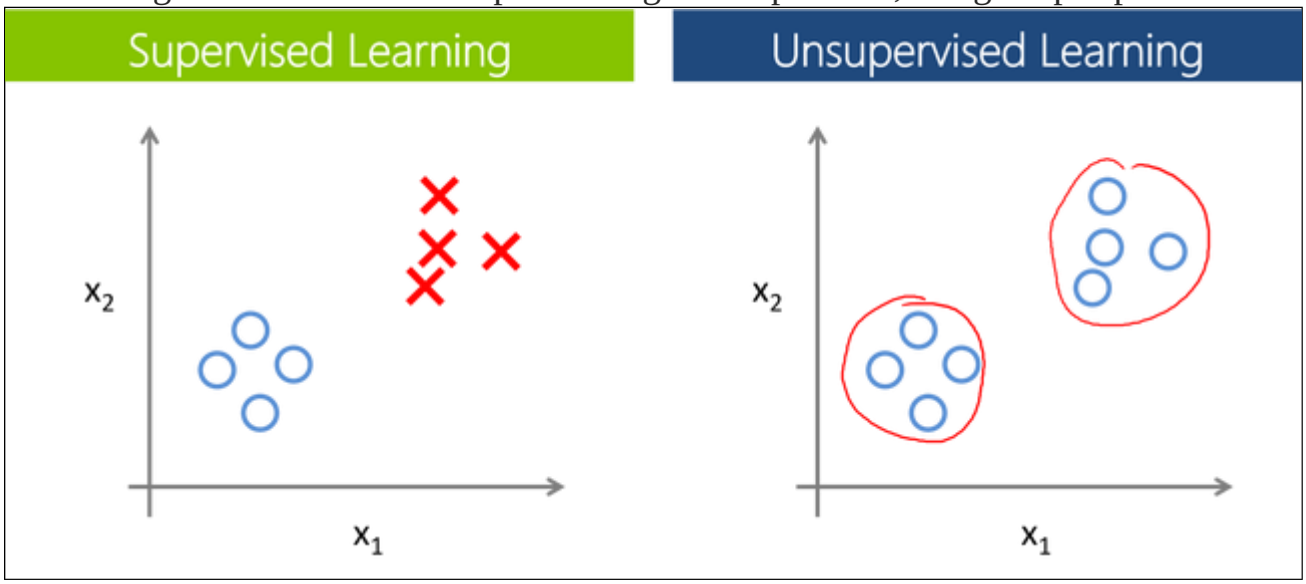
3. **Filling the missing values:** Whenever we encounter missing data in the data set then we can fill the missing data manually, most commonly the mean, median or highest frequency value is used.

4. **Machine learning:** If we have some missing data then we can predict what data shall be present at the empty position by using the existing data.

5. **Outliers detection:** There are some error data that might be present in our data set that deviates drastically from other observations in a data set. [Example: human weight = 800 Kg; due to mistyping of extra 0]

4.3. Researching the model that will be best for the type of data

Our main goal is to train the best performing model possible, using the pre-processed data.



Supervised Learning:

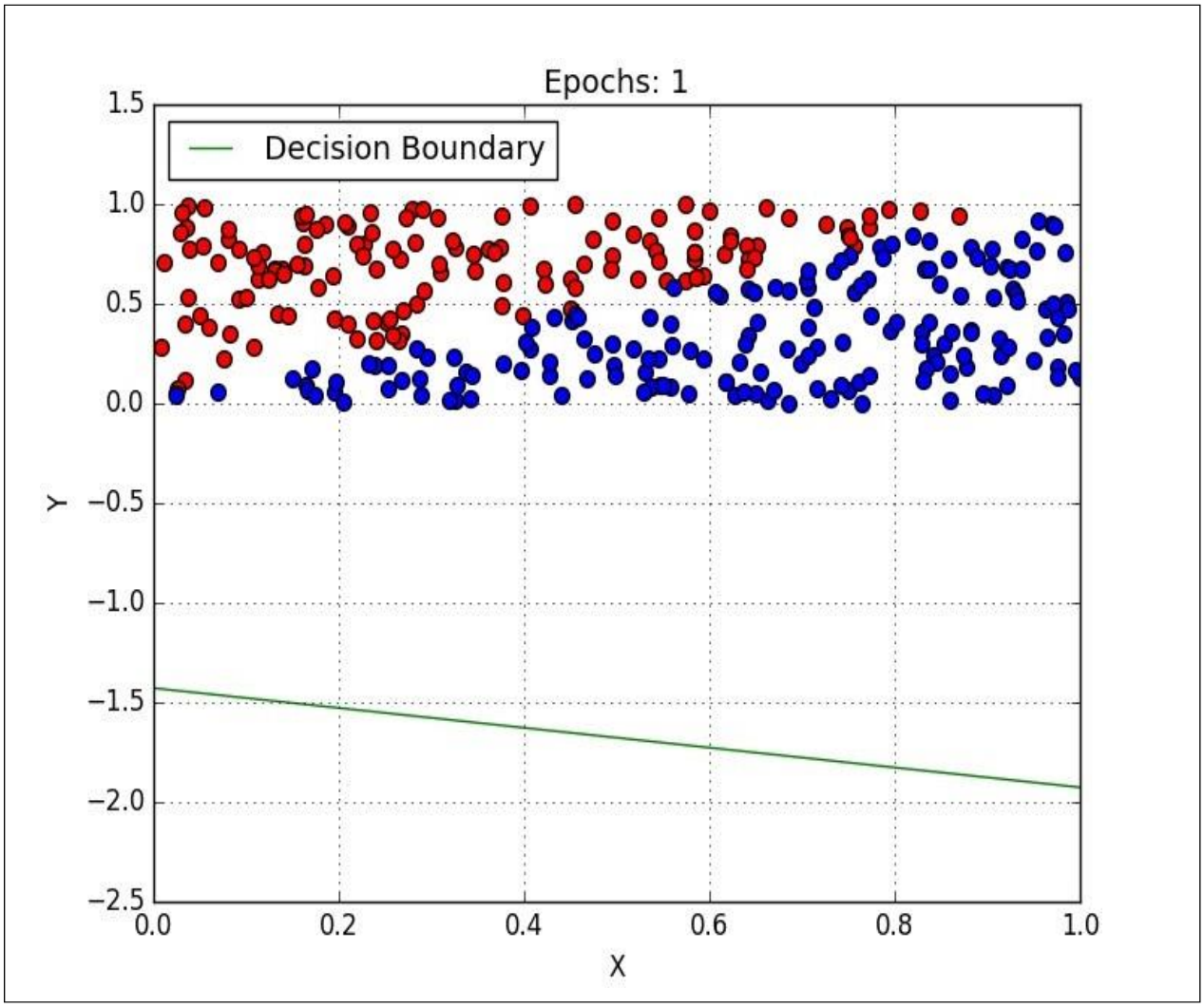
In Supervised learning, an AI system is presented with data which is labelled, which means that each data tagged with the correct label.

The supervised learning is categorized into 2 other categories which are "**Classification**" and "**Regression**".

Classification:

Classification problem is when the target variable is **categorical** (i.e. the output could be classified into classes — it belongs to either Class A or B or something else).

A classification problem is when the output variable is a category, such as “red” or “blue” , “disease” or “no disease” or “spam” or “not spam”.



As shown in the above representation, we have 2 classes which are plotted on the graph i.e. red and blue which can be represented as ‘setosa flower’ and ‘versicolor flower’, we can image the X-axis as ther ‘Sepal Width’ and the Y-axis as the ‘Sepal Length’, so we try to create the best fit line that separates both classes of flowers.

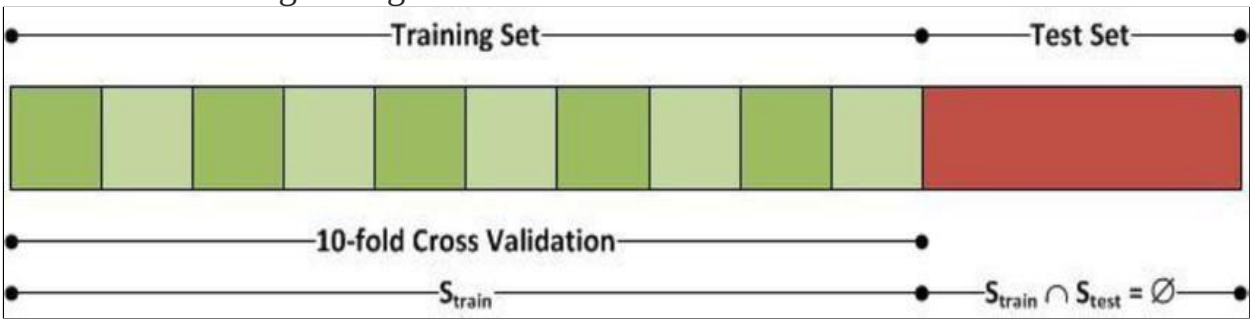
These some most used classification algorithms.

- **K-Nearest Neighbor**
- **Naive Bayes**
- **Decision Trees/Random Forest**
- **Support Vector Machine**
- **Logistic Regression**

4.4. Training and testing the model on data

For training a model we initially split the model into 3 three sections which are '**Training data**','**Validation data**' and '**Testing data**'.

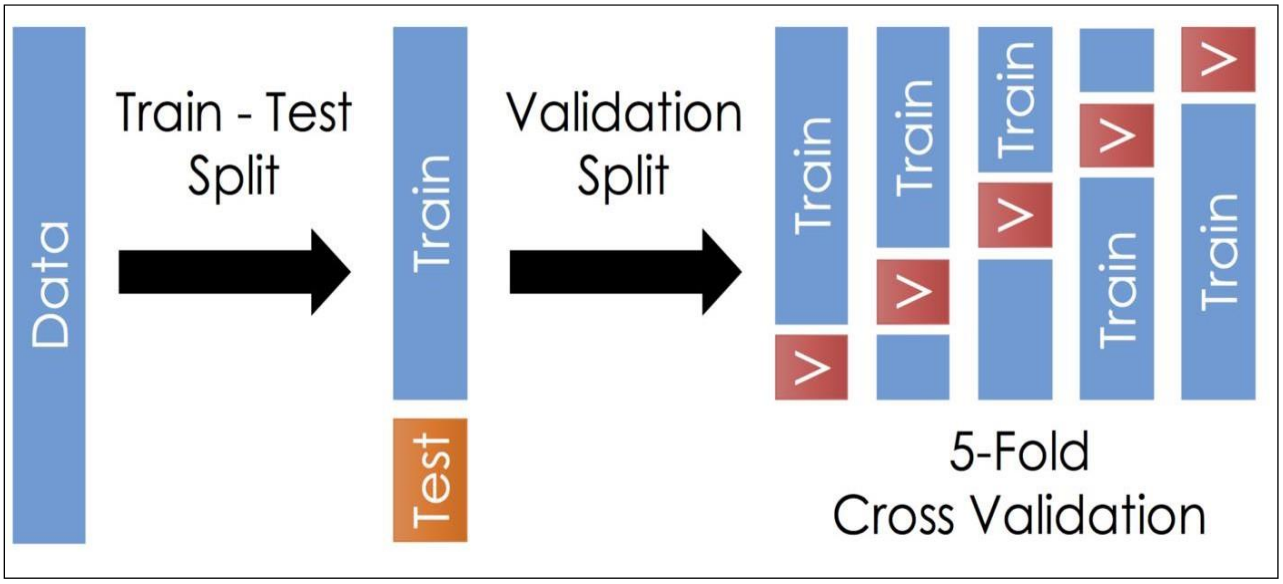
You train the classifier using '**training data set**', tune the parameters using '**validation set**' and then test the performance of your classifier on unseen '**test data set**'. An important point to note is that during training the classifier only the training and/or validation set is available. The test data set must not be used during training the classifier. The test set will only be available during testing the classifier.



Training set: The training set is the material through which the computer learns how to process information. Machine learning uses algorithms to perform the training part. A set of data used for learning, that is to fit the parameters of the classifier.

Validation set: Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. A set of unseen data is used from the training data to tune the parameters of a classifier.

Test set: A set of unseen data used only to assess the performance of a fully-specified classifier.



Once the data is divided into the 3 given segments we can start the training process.

In a data set, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. Usually, a data set is divided into a training set, a validation set (some people use ‘test set’ instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration.

The model uses any one of the models that we had chosen in step 3/ point 3. Once the model is trained we can use the same trained model to predict using the testing data i.e. the unseen data. Once this is done we can develop a confusion matrix, this tells us how well our model is trained. A confusion matrix has 4 parameters, which are '**True positives'**, **'True Negatives'**, **'False Positives'** and '**False Negative'**. We prefer that we get more values in the True negatives and true positives to get a more accurate model. The size of the Confusion matrix completely depends upon the number of classes.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

- **True positives:** These are cases in which we predicted TRUE and our predicted output is correct.
- **True negatives:** We predicted FALSE and our predicted output is correct.
- **False positives:** We predicted TRUE, but the actual predicted output is FALSE.
- **False negatives:** We predicted FALSE, but the actual predicted output is TRUE.

We can also find out the accuracy of the model using the confusion matrix.

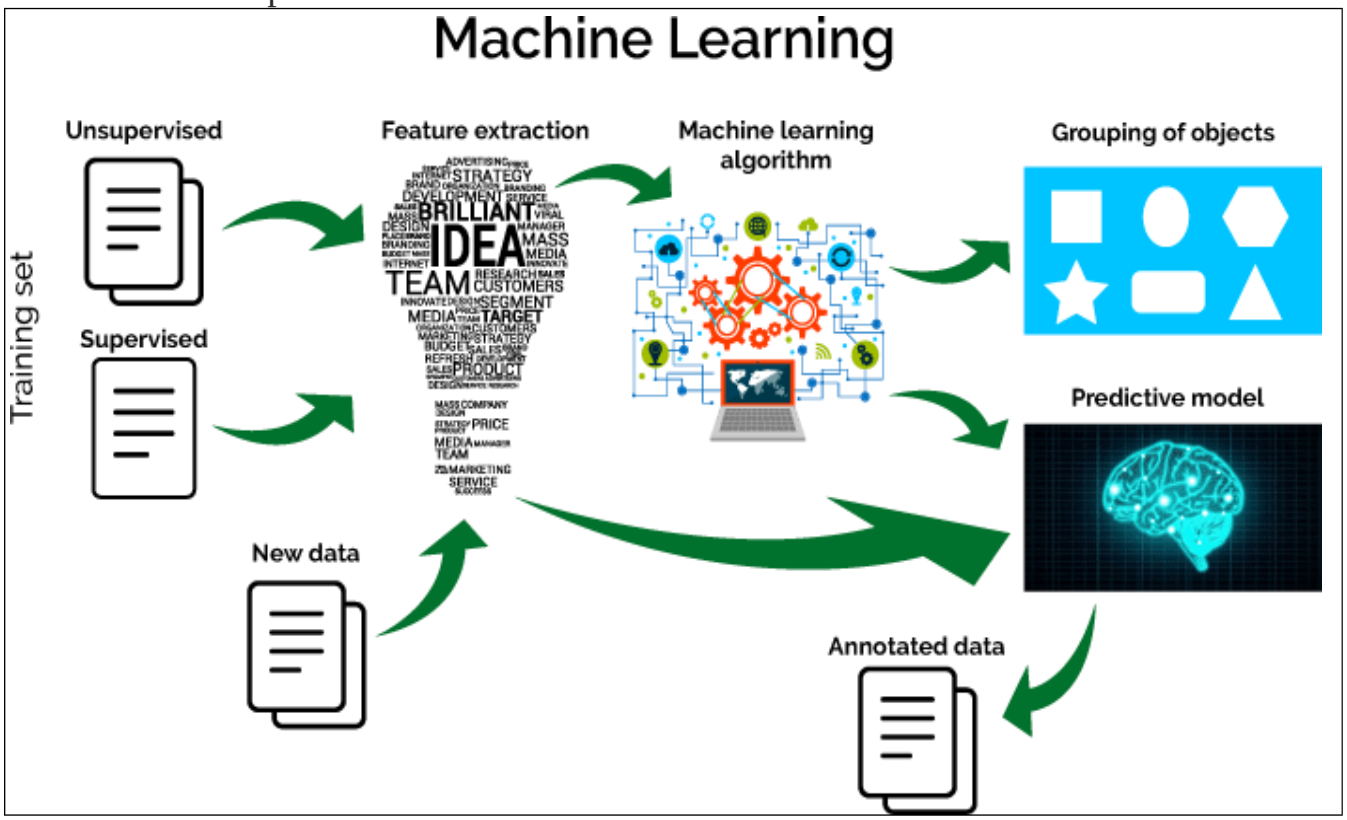
Accuracy = (True Positives +True Negatives) / (Total number of classes)

i.e. for the above example:

Accuracy = (100 + 50) / 165 = 0.9090 (90.9% accuracy)

4.5. Evaluation

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future.



To improve the model, we might tune the hyper-parameters of the model and try to improve the accuracy and also looking at the confusion matrix to try to increase the number of true positives and true negatives.

APPENDIX

Model Selection

```
y = df['PlacedOrNot']
X = df.drop(['PlacedOrNot','AgeBand'], axis=1)
```

Importing Libraries

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,
plot_confusion_matrix,ConfusionMatrixDisplay

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
models_accuracy = {}
cv = KFold(n_splits=15, random_state=13, shuffle=True)
```

1 . Logistic Regression

```
logr = LogisticRegression(solver='liblinear')
logr.fit(X_train, y_train)
logr_score = logr.score(X_test, y_test)
models_accuracy['Logistic Regression'] = logr_score*100

logr_score*100
```

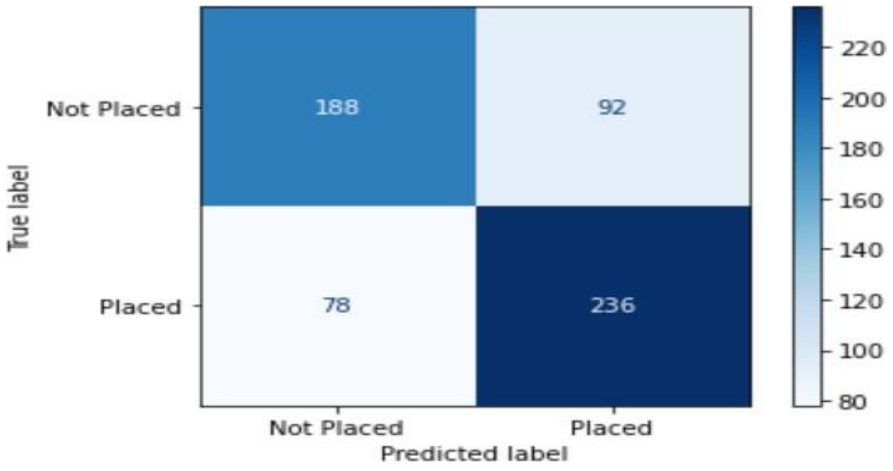
Classification Report for Logistic Regression

```
pred = logr.predict(X_test)
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	0.71	0.67	0.69	280
1	0.72	0.75	0.74	314
accuracy			0.71	594
macro avg	0.71	0.71	0.71	594
weighted avg	0.71	0.71	0.71	594

Confusion Matrix

```
ConfusionMatrixDisplay.from_estimator(logr, X_test, y_test cmap =
plt.cm.Blues,display_labels = ['Not Placed', 'Placed'])
plt.grid(False)
plt.show();
```



2 . Support Vector Machine

```
svm_model = SVC(decision_function_shape='ovr')
svm_model.fit(X_train, y_train)
svm_score = svm_model.score(X_test, y_test)

models_accuracy['SVM'] = svm_score*100
svm_score*100
```

Classification Report for Support Vector Machine

```
pred = svm_model.predict(X_test)
print(classification_report(y_test, pred))
```

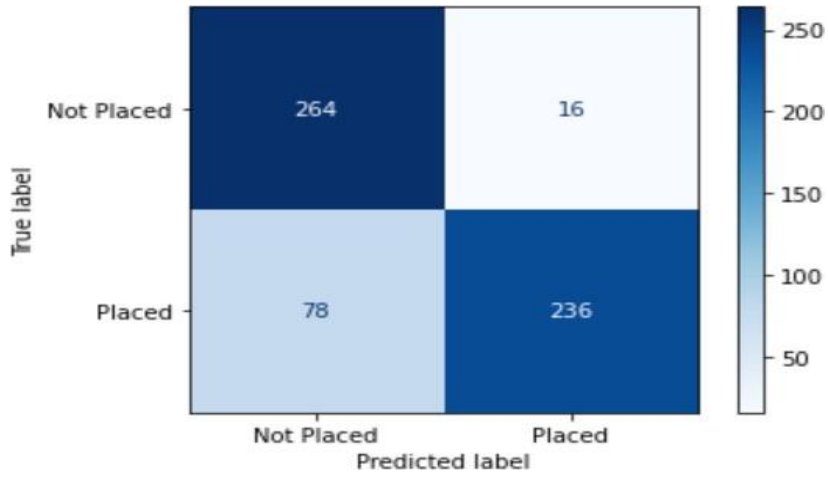
```
                           precision    recall  f1-score   support

                        0       0.77      0.94      0.85       280
                        1       0.94      0.75      0.83       314

                 accuracy                           0.84       594
                macro avg       0.85      0.85      0.84       594
             weighted avg       0.86      0.84      0.84       594
```

Confusion Matrix

```
ConfusionMatrixDisplay.from_estimator(svm_model, X_test, y_test,
cmap = plt.cm.Blues,display_labels = ['Not Placed', 'Placed'])
plt.grid(False)
plt.show();
```



3 . KNeighborsClassifier

```
kn_model = KNeighborsClassifier(n_neighbors=15)
kn_model.fit(X_train, y_train)
kn_score = kn_model.score(X_test, y_test)
models_accuracy['Knn'] = kn_score*100
kn_score*100
```

Classification Report for Support Vector Machine

```
pred = kn_model.predict(X_test)
print(classification_report(y_test, pred))
```

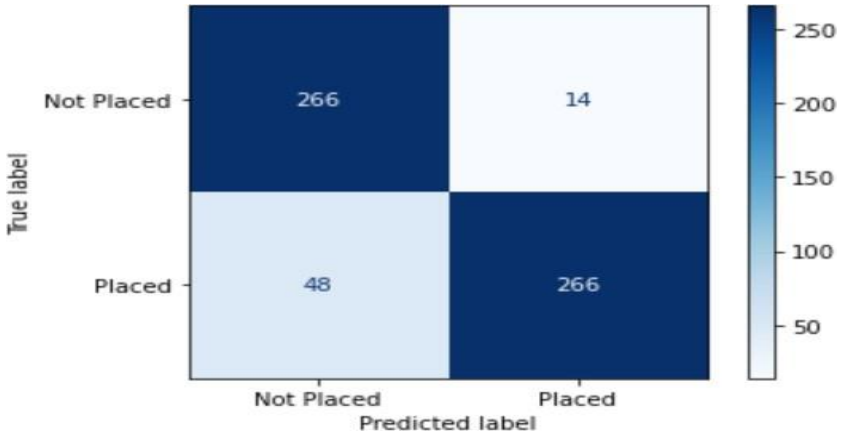
```
                precision    recall  f1-score   support

           0       0.81      0.95      0.88       280
           1       0.95      0.81      0.87       314

    accuracy                           0.88       594
   macro avg       0.88      0.88      0.88       594
weighted avg       0.89      0.88      0.88       594
```

Confusion Matrix

```
ConfusionMatrixDisplay.from_estimator(ran_model, X_test, y_test,
cmap = plt.cm.Blues,display_labels = ['Not Placed', 'Placed'])
plt.grid(False)
plt.show();
```



4. RandomForestClassifier

```
ran_model = RandomForestClassifier(n_estimators = 40)
ran_model.fit(X_train, y_train)
ran_score = ran_model.score(X_test, y_test)
models_accuracy['RanForest'] = ran_score*100

ran_score*100
```

Classification Report for Support Vector Machine

```
pred = ran_model.predict(X_test)
print(classification_report(y_test, pred))
```

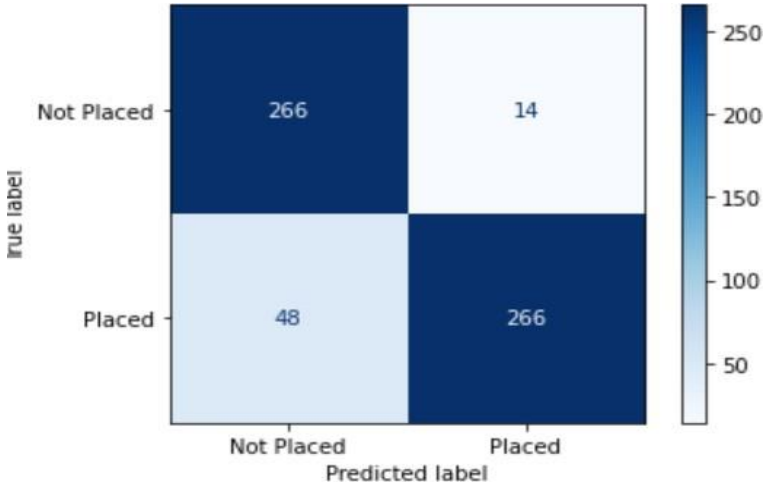
```
                            precision    recall  f1-score   support

                         0       0.85      0.95      0.90       280
                         1       0.95      0.85      0.90       314

                  accuracy                           0.90       594
                 macro avg       0.90      0.90      0.90       594
              weighted avg       0.90      0.90      0.90       594
```

Confusion Matrix

```
ConfusionMatrixDisplay.from_estimator(ran_model,
X_test, y_test,cmap = plt.cm.Blues, display_labels = ['Not Placed',
'Placed'])
plt.grid(False)
plt.show();
```



5. AdaBoostClassifier

```
ada_model = AdaBoostClassifier()
ada_model.fit(X_train, y_train)
ada_score = ada_model.score(X_test, y_test)
models_accuracy['AdaBoost'] = ada_score*100

ada_score*100
```

Classification Report for Support Vector Machine

```
pred = ada_model.predict(X_test)
print(classification_report(y_test, pred))
```

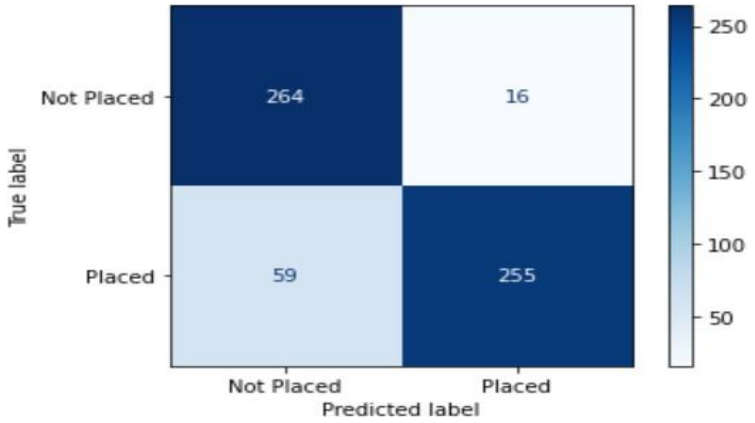
```
              precision    recall  f1-score   support

           0       0.82      0.94      0.88       280
           1       0.94      0.81      0.87       314

    accuracy                           0.87       594
   macro avg       0.88      0.88      0.87       594
weighted avg       0.88      0.87      0.87       594
```

Confusion Matrix

```
ConfusionMatrixDisplay.from_estimator(ada_model, X_test, y_test,cmap =
plt.cm.Blues,display_labels = ['Not Placed', 'Placed'])
plt.grid(False)
plt.show();
```



6. GradientBoostingClassifier

```
grad_model = GradientBoostingClassifier()
grad_model.fit(X_train, y_train)
grad_score = grad_model.score(X_test, y_test)
models_accuracy['AdaBoost'] = grad_score*100

grad_score*100
```

Classification Report for Support Vector Machine

```
pred = grad_model.predict(X_test)
print(classification_report(y_test, pred))
```

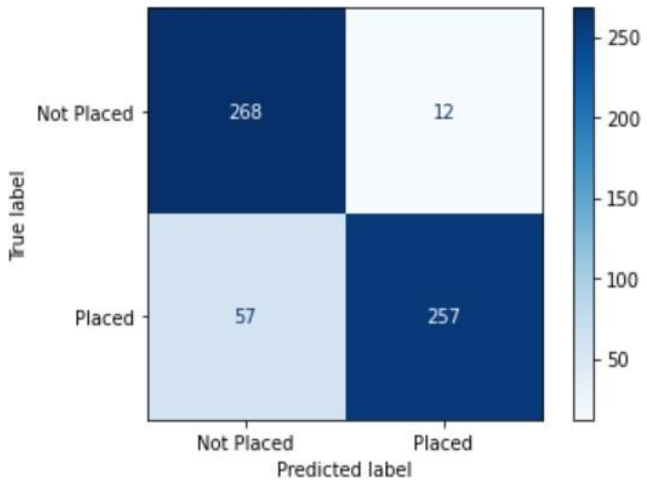
```
                           precision    recall  f1-score   support

                        0       0.82      0.96      0.89       280
                        1       0.96      0.82      0.88       314

                 accuracy                           0.88       594
                macro avg       0.89      0.89      0.88       594
             weighted avg       0.89      0.88      0.88       594
```

Confusion Matrix

```
ConfusionMatrixDisplay.from_estimator(grad_model, X_test, y_test,cmap =
plt.cm.Blues,display_labels = ['Not Placed', 'Placed'])
plt.grid(False)
plt.show();
```



7. DecisionTreeClassifier

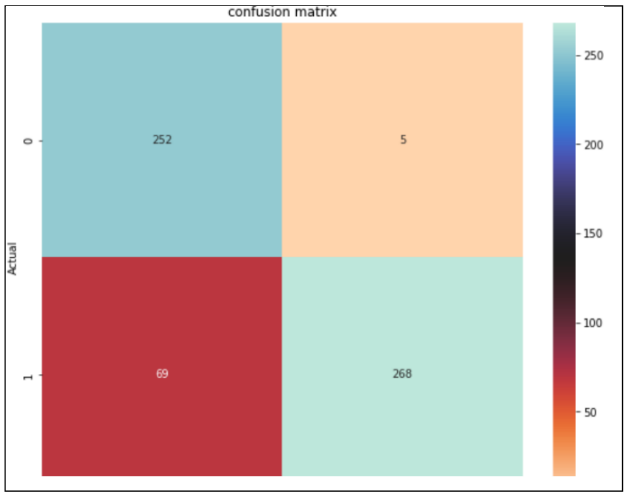
```
from sklearn.tree import DecisionTreeClassifier
c1f=DecisionTreeClassifier(criterion='gini',max_depth=7)
c1f=c1f.fit(X_train,Y_train)
```

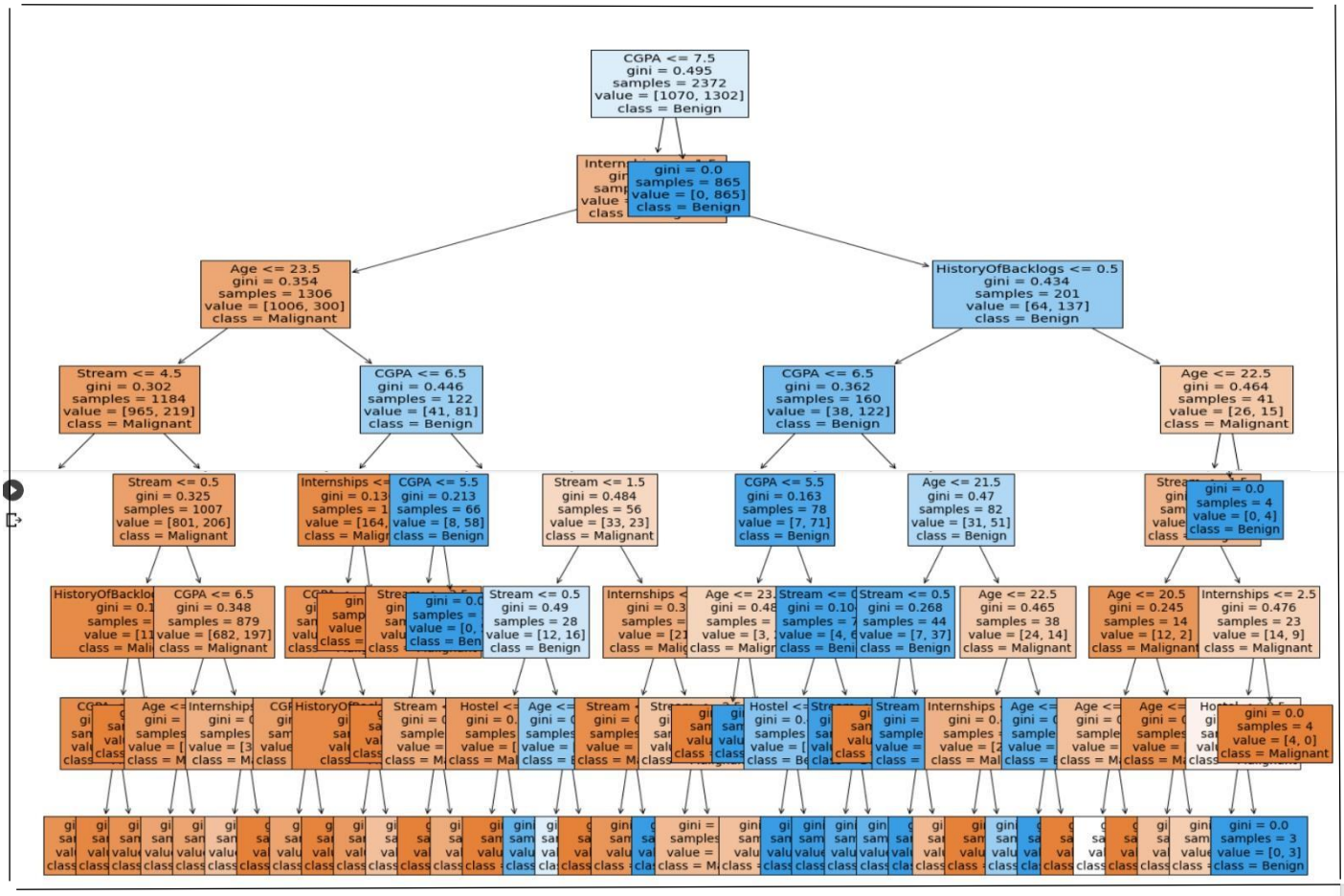
Classification Report for Support Vector Machine

```
c1f.predict_proba(X_test)
array([[0. , 1. ], [0. , 1. ],
       [0.66521739, 0.33478261], ...,
       [0.66521739, 0.33478261],
       [0.78354978, 0.21645022],
       [0. , 1. ]])
```

Confusion Matrix

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,predictions,labels=[0,1])
import seaborn as sns
ax = sns.heatmap(cm,cmap="icefire_r", annot=True, fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('confusion mastrix')
plt.show()
```




CGPA <= 7.5
gini = 0.495
samples = 2372
value = [1070, 1302]
class = Benign
gini = 0.0
samples = 865
value = [0, 865]
class = Benign
Age <= 23.5
gini = 0.354
samples = 1306
value = [1006, 300]
class = Malignant
HistoryOfBacklogs <= 0.5
gini = 0.434
samples = 201
value = [64, 137]
class = Benign
Stream <= 4.5
gini = 0.302
samples = 1184
value = [965, 219]
class = Malignant
CGPA <= 6.5
gini = 0.446
samples = 122
value = [41, 81]
class = Benign
CGPA <= 6.5
gini = 0.362
samples = 160
value = [38, 122]
class = Benign
Age <= 22.5
gini = 0.464
samples = 41
value = [26, 15]
class = Malignant
Stream <= 0.5
gini = 0.325
samples = 1007
value = [801, 206]
class = Malignant
Stream <= 1.5
gini = 0.484
samples = 56
value = [33, 23]
class = Malignant
CGPA <= 5.5
gini = 0.163
samples = 78
value = [7, 71]
class = Benign
Age <= 21.5
gini = 0.47
samples = 82
value = [31, 51]
class = Benign
gini = 0.0
samples = 4
value = [0, 4]
class = Benign
HistoryOfBacklo
gini = 0.1
samples =
value = [11
class = Mali
CGPA <= 6.5
gini = 0.348
samples = 879
value = [682, 197]
class = Malignant
Stream <= 0.5
gini = 0.49
samples = 28
value = [12, 16]
class = Benign
Age <= 22.5
gini = 0.465
samples = 38
value = [24, 14]
class = Malignant
Age <= 20.5
gini = 0.245
samples = 14
value = [12, 2]
class = Malignant
Internships <= 2.5
gini = 0.476
samples = 23
value = [14, 9]
class = Malignant
gini = 0.0
samples = 4
value = [0, 4]
class = Malignant
gini = 0.0
samples = 3
value = [0, 3]
class = Benign

CONCLUSION

We have tested the various Algorithm models and comparing their Confusion Matrix in the below Tabular Column.

N=594	True Positive	True Negative	False Positive	False Negative	Accuracy Score
Logistic Regression	188	236	78	92	71.38047138047138
Support Vector Matrix	264	236	78	16	84.17508417508418
KNeighbours Classifier	266	266	48	14	87.54208754208754
Random Forest Classifier	266	266	40	14	89.56228956228956
AdaBoost Classifier	264	255	59	16	87.37373737373737
Gradient Boosting Classifier	268	257	57	12	88.38383838383838
Decision Tree Classifier	252	268	69	5	87.54208754208754

N=594	Precision	Recall
Logistic Regression	0.7067	0.6714
Support Vector Matrix	0.7719	0.9428
KNeighbours Classifier	0.8471	0.95
Random Forest Classifier	0.86	0.95
AdaBoost Classifier	0.8173	0.942
Gradient Boosting Classifier	0.795	0.957
Decision Tree Classifier	0.9805	0.7850

Results and discussions

- In comparision with all the classifier Random Forest Classifier gives the highest accuracy score of 89.562 .
- Gradient Boosting method makes the highest True positive prediction 268 out of 594 rows of test data.
- Decision Tree classifier makes the highest True negative prediction 268 out of 594 rows of test data.
- Initially we had got an accuracy around 82.51% .After adding three more attributes we were able to get an accuracy of 89.562

Student Placement Predictor is a system which predicts student placement status using machine learning techniques.

Many research papers are there related to educational sector, all these papers mainly concentrate on student performance predictions. All these predictions help the institute to improvise the student performance and can come up with 100% results.

Many of the previous research papers concentrate on a less number of parameters such as CGPA and Arrears for placement status prediction which leads to less accurate results, but proposed work contains many educational parameters to predict placement status which will be more accurate

REFERENCES

[1]. Mangasuli Sheetal B, Prof. Savita Bakare "Prediction of Campus Placement Using Data Mining AlgorithmFuzzy logic and K nearest neighbour" International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 6, June 2016 .

[2] "Prediction Model for Students Future Development by Deep Learning and TensorFlow Artificial Intelligence Engine" 2018 4th IEEE International Conference on Information Management.

[3]. K. Sreenivasa Rao, N. Swapna, P. Praveen Kumar Educational data mining for student placement prediction using machine Learning algorithms Research Paper, International Research Journal of Engineering and Technology (IRJIET) 2018

[4]. Ankita A Nichat, Dr.Anjali B Raut "Predicting and Analysis of Student Performance Using Decision Tree Technique" International Journal of Innovative Research in Computer and Communication Engineering V0l. 5, Issue 4, April 2017.

[5]. Oktariani Nurul Pratiwi "Predicting Student Placement Class using Data Mining" IEEE International Conference 2013.

[6]. Ajay Kumar Pal and Saurabh Pal, "Classification Model of Prediction for Placement of Students", I. J. Modern Education and Computer Science, 2013, 11, 49-56