GRADUATE ADMISSION ANALYSID AND PREDICTION

A MINI PROJECT

Submitted by

KODURI SAI MANAS HARSHA VARDHAN (RA2111027010079)

J BHARAT KESAV (RA2111027010090)

K SAI VISHAL (RA2111027010075)

Under the guidance of

Dr. E. Sasikala

Professor

Department of Data Science and Business Systems

In partial fulfilment for the

Course of

18CSE392T- Machine Learning-I

in

Department of Data Science and Business Systems

SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603203

October 2023

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this mini project report "**Graduate Admission Analysis and Prediction**" is the Bonafide work of **KODURI SAI MANAS HARSHA VARDHAN (RA2111027010079), JBHARAT KESAV(RA2111027010090)** and **K SAI VISHAL(RA2111027010075)** who carried out the project work under my supervision.

Dr. E. Sasikala
Professor
Department of Data Science and Business Systems
SRM institute of science and technology

Dr. M Lakshmi
Professor & HOD
Department of DSBS
SRM institute of science and technology

ABSTRACT

With the increase in the number of graduates who wish to pursue their education, it becomes more challenging to get admission to the students' dream university. Newly graduate students usually are not knowledgeable of the requirements and the procedures of the postgraduate admission and might spent a considerable amount of money to get advice from consultancy organizations to help them identify their admission chances. However, giving the limited number of universities that can be considered by a human consultant, this approach might be bias and inaccurate. Thus, in this paper, a machine learning approach is developed to automatically predict the possibility of postgraduate admission to help graduates recognizing and targeting the universities which are best suitable for their profile. This paper evaluates learning strategies of regression to predict the university rate given the students' profile; namely, linear regression, decision tree, and logistic regression, random forest,k-means algoritm,clustering model. This paper evaluates, these models to select the best model in terms of the highest accuracy rate and the least error. Logistic Regression model shows the most accurate prediction in our experiments, and hence, we suggest employing this model to predict the future applicant's universitychanceofadmission.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

CSS	Cascading Style Sheet
DB	Data Base
ER	Entity Relationship
SQL	Structured Query Language
HTML	Hyper Text Markup Language
UI	User Interface

OBJECTIVE

Aim:

To determine the graduate admission analysis and prediction using machine learning algorithms.

Synopsis:

Student admission problem is very important in educational institutions. This paper addresses machine learning models to predict the chance of a student to be admitted to a master's program. This will assist students to know in advance if they have a chance to get accepted. The machine learning models are multiple linear regression, k-nearest neighbor, random forest.

REQUIREMENT SPECIFICATIONS

INTRODUCTION

The world markets are developing rapidly and continuously looking for the best knowledge and experience among people. Young workers who want to stand out in their jobs are always looking for higher degrees that can help them in improving their skills and knowledge. As a result, the number of students applying for graduate studies has increased in the last decade. This fact has motivated us to study the grades of students and the possibility of admission for master's programs that can help universities in predicting the possibility of accepting master's students submitting each year and provide the needed resources.

The dataset presented in this paper is related to educational domain. Admission is a dataset with 500 rows that contains 7 different independent variables which are:

Graduate Record Exam[1] (GRE) score. The score will be out of 340 points.

Test of English as a Foreigner Language[2] (TOEFL) score, which will be out of 120 points.

• University Rating (Uni.Rating) that indicates the Bachelor University ranking among the other universities. The score will be out of 5. Statement of purpose (SOP) which is a document written to show the candidate's life, ambitious and the motivations for the chosen degree/ university. The score will be out of 5 points.

Letter of Recommendation Strength (LOR) which verifies the candidate professional experience, builds credibility, boosts confidence and ensures your competency. The score is out of 5 points. Undergraduate GPA (CGPA) out of 10. One dependent variable can be predicted which is chance of admission, that is according to the input given will be ranging from 0 to 1.

HARDWARE AND SOFTWARE SPECIFICATION

HARDWARE REQUIREMENTS

- Hard disk : 500 GB and above.
- Processor : i3 and above.
- Ram : 4GB and above.

SOFTWARE REQUIREMENTS

- Operating System : Windows 10
- Software : python
- Tools : Anaconda (Jupiter Notebook IDE)

TECHNOLOGIES USED

- Programming Language : **Python**

INTRODUCTION TO PYTHON

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).
- Python has a simple syntax like the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way, or a functional way.

Good to know.

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.
- Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End of Life date of January 1, 2020, has been established for Python 2, after which time it will no longer be maintained.
- Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator for Life) by the 12 Python community. The name Python derives not from the snake, but from the British comedy troupe Monty Python’s Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
- It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages.

- Python was designed to for readability and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope, such as the scope of loops, functions, and classes. Other programming languages often use curly brackets for this purpose.

Python is Interpreted

- ● Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.
- ● This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
- ● One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
- ● In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The 13 expediency of coding in an interpreted language is typically worth it for most applications.
- ● For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.
- ● Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.
- ● Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

Machine learning

Introduction:

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task.
Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine learning tasks:

1. Machine learning encompasses a wide range of tasks and applications, each serving different purposes across various domains. Here are some common machine learning tasks:
2. Supervised Learning:
3. Classification: Assigns input data to predefined categories or classes.
4. Regression: Predicts a continuous outcome or value based on input features.
5. Unsupervised Learning:
6. Clustering: Groups similar data points together without predefined categories.
7. Dimensionality Reduction: Reduces the number of input features while retaining important information.
    - Association: Discovers patterns and relationships between variables in large datasets.

8. Semi-Supervised Learning:
    - Combines elements of supervised and unsupervised learning, using both labeled and unlabeled data.
9. Reinforcement Learning:
    - Involves an agent learning to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties.
10. Natural Language Processing (NLP):
    - Text Classification: Categorizes text into predefined classes or labels.
    - Named Entity Recognition (NER): Identifies entities such as names, locations, and organizations in text.
    - Machine Translation: Translates text from one language to another.
    - Sentiment Analysis: Determines the sentiment expressed in text (positive, negative, neutral).
11. Computer Vision:
    - Image Classification: Assigns labels to images based on their content.
    - Object Detection: Identifies and locates objects within images.
    - Facial Recognition: Recognizes and verifies individuals based on facial features.
12. Anomaly Detection:
    - Identifies unusual patterns or outliers in data that may indicate errors, fraud, or other anomalies.
13. Recommendation Systems:
    - Provides personalized suggestions or recommendations based on user preferences and behavior.
14. Time Series Analysis:
    - Analyzes data collected over time to make predictions or identify patterns.
15. Transfer Learning:
    - Utilizes knowledge gained from one task to improve performance on a different but related task.
16. Ensemble Learning:
    - Combines predictions from multiple models to improve overall performance and generalization.
17. Generative Models:
    - Creates new data instances that resemble a given dataset, often used in tasks like image generation.

These tasks represent just a subset of the diverse applications of machine learning. The choice of task depends on the nature of the problem and the type of data available.

Types of learning algorithms:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

Supervised learning:

Supervised learning is a type of machine learning where the algorithm is trained on a labeled dataset, meaning that the input data used for training is paired with corresponding output labels. The goal of supervised learning is to learn a mapping or relationship between the input features and the target labels so that, when presented with new, unseen data, the algorithm can make accurate predictions or classifications.

In supervised learning, the algorithm learns from the labeled examples in the training dataset and generalizes its knowledge to make predictions on new, unseen data. The process typically involves the following steps:

Input Data: The dataset consists of input features and their corresponding output labels. The input features are the characteristics or attributes of the data, and the output labels are the desired predictions or classifications.

Training: The algorithm is trained on the labeled dataset, adjusting its internal parameters to minimize the difference between its predictions and the true labels. The objective is to learn a mapping function that can accurately predict the output labels for new, unseen inputs.

Validation: The model's performance is evaluated on a separate validation dataset that it has not seen during training. This step helps assess how well the model generalizes to new data and whether it is overfitting (memorizing the training data) or underfitting (failing to capture the underlying patterns).

Testing: Once the model has been trained and validated, it is tested on a completely independent test dataset to evaluate its performance in a real-world scenario.

Supervised learning algorithms can be broadly categorized into two main types:

Classification: The algorithm predicts a discrete category or label. For example, classifying emails as spam or not spam, or identifying whether an image contains a cat or a dog.

Regression: The algorithm predicts a continuous value. For instance, predicting the price of a house based on its features or estimating the temperature based on various environmental factors.

Common supervised learning algorithms include:

Linear Regression
Logistic Regression
Support Vector Machines (SVM)
Decision Trees and Random Forests
K-Nearest Neighbors (KNN)
Neural Networks (Deep Learning)
Gradient Boosting Algorithms (e.g., XGBoost, LightGBM)
Naive Bayes
The choice of algorithm depends on the nature of the problem, the characteristics of the data, and the desired outcome (classification or regression).

Unsupervised learning:
Unsupervised learning is a type of machine learning where the algorithm is trained on a dataset without explicit supervision, meaning that there are no labeled output variables to guide the learning process. The system tries to learn the patterns and structure from the input data without being explicitly told what to look for.

In unsupervised learning, the algorithm explores the inherent structure and relationships within the data. The primary goal is often to discover hidden patterns, group similar data points together, reduce dimensionality, or otherwise extract meaningful insights from the data.

Common unsupervised learning algorithms include:

Clustering Algorithms:

K-Means: Divides the data into k clusters based on similarity.
Hierarchical Clustering: Builds a tree of clusters, where the leaves are individual data points.
Dimensionality Reduction Algorithms:

Principal Component Analysis (PCA): Reduces the dimensionality of the data while retaining important information.
t-Distributed Stochastic Neighbor Embedding (t-SNE): Visualizes high-dimensional data in two or three dimensions.
Association Rule Learning:

Apriori Algorithm: Discovers relationships between variables in large datasets, often used in market basket analysis.
Generative Models:

Variational Autoencoders (VAEs): Learn a probabilistic mapping between the data space and latent space.
Generative Adversarial Networks (GANs): Generate new data instances that resemble the training data.
Anomaly Detection:

Algorithms like Isolation Forests or One-Class SVM identify instances that deviate significantly from the norm.
Unsupervised learning is particularly useful when the goal is to explore the inherent structure of data, discover patterns, or preprocess data for subsequent tasks. It is often employed in scenarios where labeled data is scarce or unavailable. However, the interpretation of results in unsupervised learning can be more challenging compared to supervised learning, as there are no explicit target labels to evaluate performance..

Semi-supervised learning:
Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

K-Nearest Neighbors

Introduction In four years of analytics built more than 80% of classification models and just 15- 20% regression models. These ratios can be generalized throughout the industry. The reason for a bias towards classification models is that most analytical problems involve making a decision. For instance, will a customer attrite or not, should we target customer X for digital campaigns, whether customer has a high potential or not etc. This analysis is more insightful and directly links to an implementation roadmap. In this article, we will talk about another widely used classification technique called K-nearest neighbors (KNN). Our focus will be primarily on how does the algorithm work and how does the input parameter effect the output/prediction.

KNN algorithm

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique, we generally look at 3 important aspects:

1. Ease to interpret output
2. Calculation time
3. Predictive Power

Decision tree

In a decision tree, the algorithm starts with a root node of a tree then compares the value of different attributes and follows the next branch until it reaches the end leaf node. It uses different algorithms to check the split and variable that allow the best homogeneous sets of population. decision trees are widely used in data science. It is a key proven tool for making decisions in complex scenarios. In Machine learning, ensemble methods like decision tree, random forest are widely used. Decision trees are a type of supervised learning algorithm where data will continuously be divided into different categories according to certain parameters. So, in this blog, I will explain the Decision tree algorithm. How is it used? How its functions will cover everything that is related to the decision tree.

What is a Decision Tree?

Decision tree as the name suggests is a flow like a tree structure that works on the principle of conditions. It is efficient and has strong algorithms used for predictive analysis. It has mainly been attributed to internal nodes, branches, and a terminal node. Every internal node holds a “test” on an attribute, branches hold the conclusion of the test, and every leaf node means the class label. This is the most used algorithm when it comes to supervised learning techniques. It is used for both classifications as well as regression. It is often termed as “CART” that means Classification and Regression Tree. Tree algorithms are always preferred due to stability and reliability.

How can an algorithm be used to represent a tree Let us see an example of a basic decision tree where it is to be decided in what conditions to play cricket and in what conditions not to play. You might have got a fair idea about the conditions on which decision trees work with the above example. Let us now see the common terms used in Decision Tree that is stated below:

- Branches - Division of the whole tree is called branches.
- Root Node - Represent the whole sample that is further divided.
- Splitting - Division of nodes is called splitting.
- Terminal Node - Node that does not split further is called a terminal node.
- Decision Node - It is a node that also gets further divided into different sub-nodes being a sub node.
- Pruning - Removal of sub nodes from a decision node.
- Parent and Child Node - When a node gets divided further then that node is termed as parent node whereas the divided nodes or the sub-nodes are termed as a child node of the parent node.

LITERATURE REVIEW

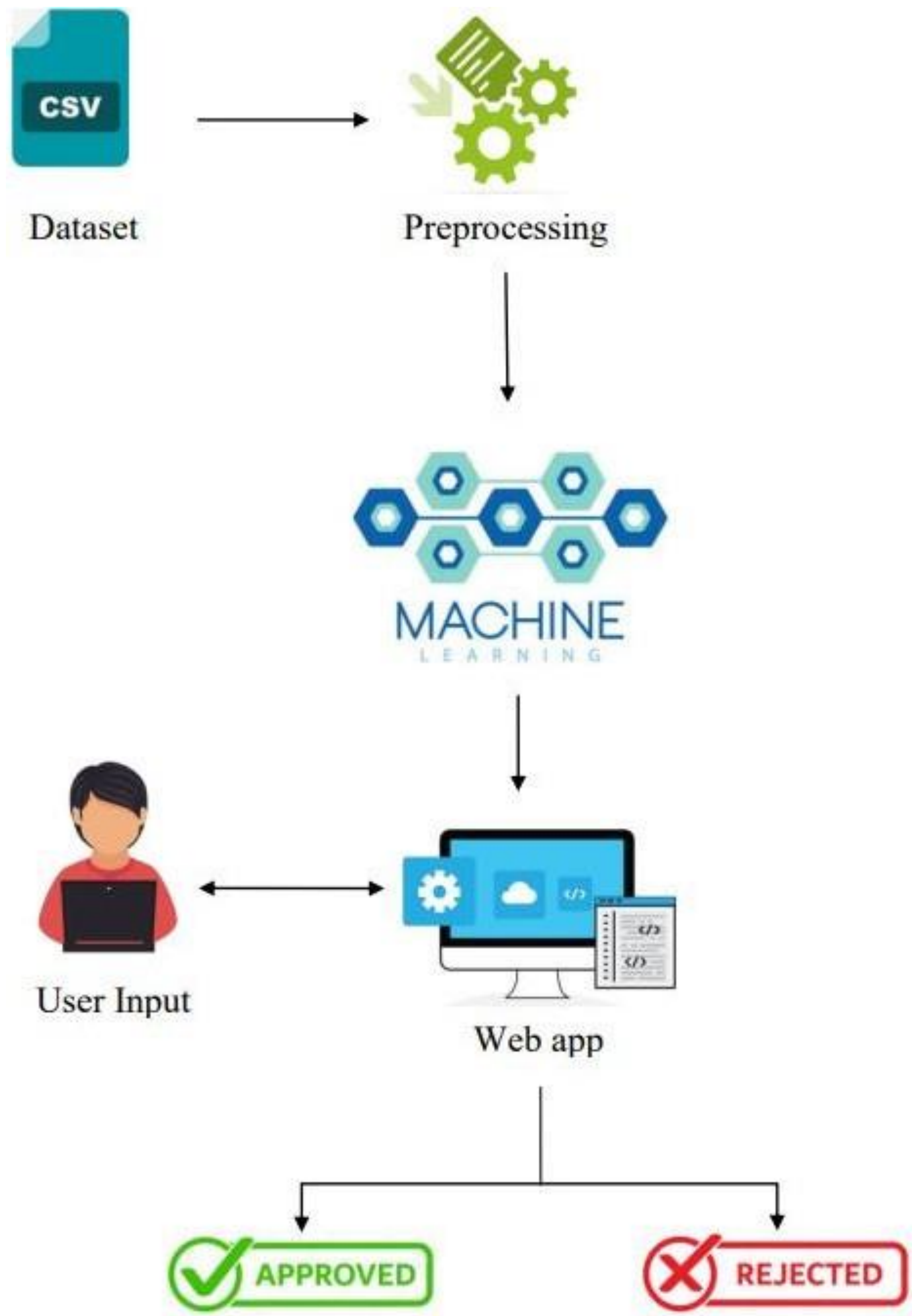
[1] "Prediction fora University Admission Using Machine Learning" by Chitra Apoorva D.A, Malepati Chandu Nath, Peta Rohith, Swaroop S, Bindushree S Blue Eyes Intelligence Engineering & Sciences Publication. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-6 March 2020.

[2] "Machine Learning Basics with the K-Nearest Neighbors Algorithm"- https://towardsdatascience.com/machine-learning- basics-with-the-k-nearest-neighbors-algorithm.

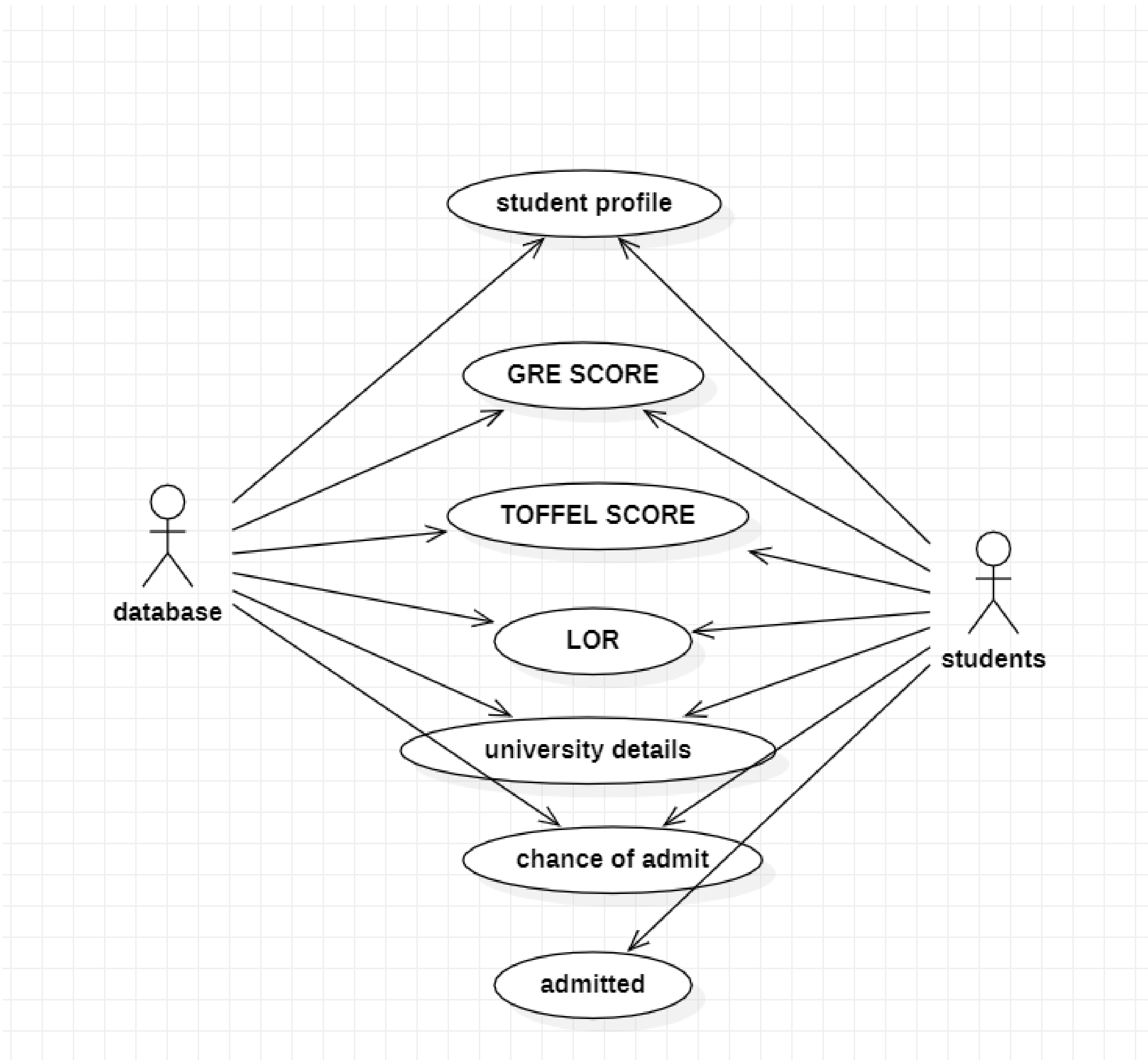
[3] "Graduate Admission Prediction Using Machine Learning" by Sara Aljasmi, Ali Bou Nassif, Ismail Shahin, Ashraf Elnagar - ResearchGate Publication, December 2020.

[4] Sujay S "Supervised Machine Learning Modelling & Analysis For Graduate Admission Prediction" Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, Volume-7 | Issue-4, August 2020.

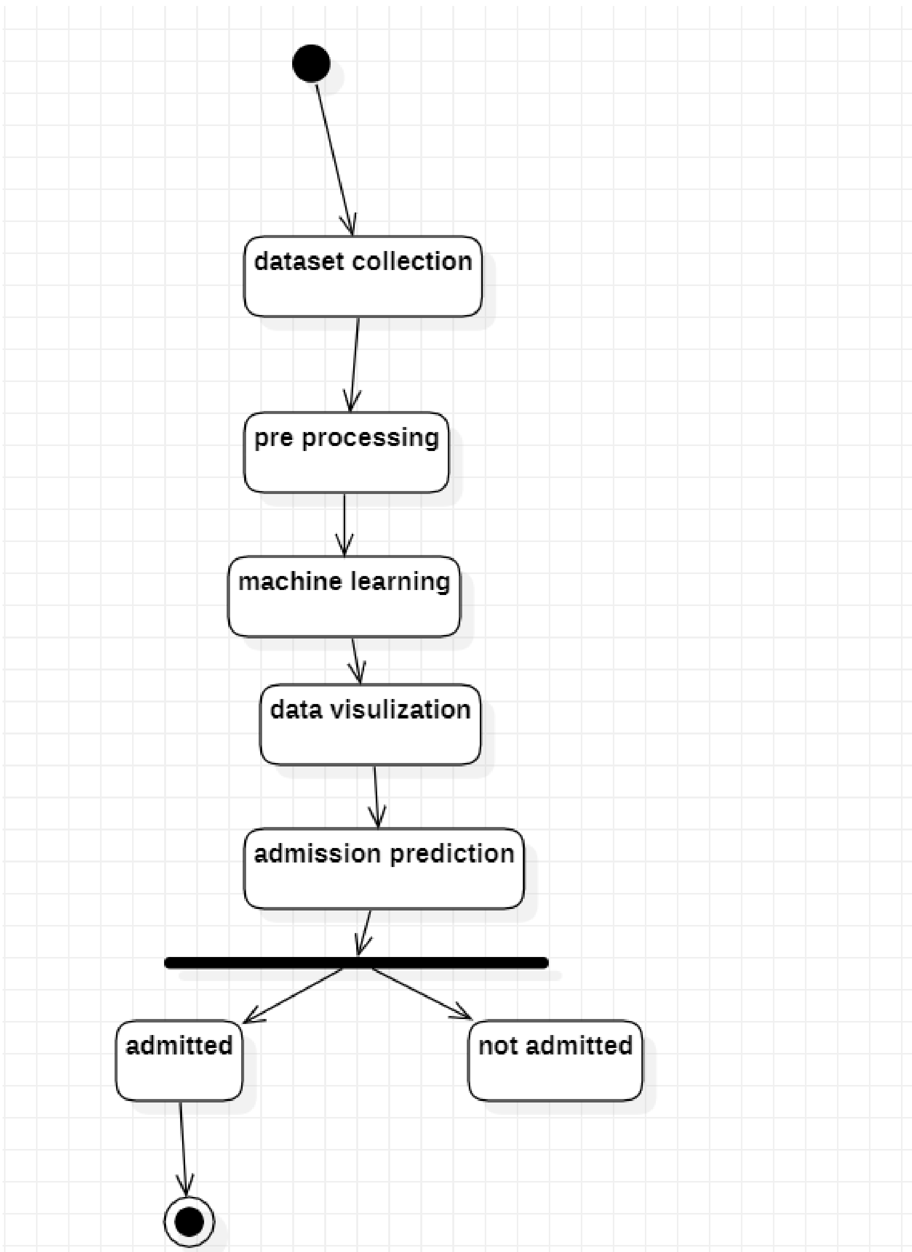
ARCHITECTURE DIAGRAM



USE CASE DIAGRAM



STATE DIAGRAM



MODULES

- ➢ Dataset collection
- ➢ Machine Learning Algorithm
- ➢ Prediction

MODULE EXPLANATION:

Dataset collection:

Dataset is collected from kaggle.com. That dataset has some value like gender, marital status, self-employed or not, monthly income, etc. Dataset has the information, whether the previous loan is approved or not depends on the customer information. That data will be preprocessed and proceed to the next step.

Machine learning Algorithm:

In this stage, the collected data will be given to the machine algorithm for the training process. We use multiple algorithms to get a high accuracy range of prediction. A preprocessed data set is processed in different machine learning algorithms. Each algorithm gives some accuracy level. Each one is undergoing for the comparison.

- ✓ Logistic Regression
- ✓ Random Forest Classifier
- ✓ Decision Tree Classifier
- ✓ SVM

SOURCE CODE:

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

Importing dataset

In [3]:
```
data = pd.read_csv("E:/graduate/Admission_Predict.csv")
data.shape
```

Out[3]: (400, 9)

Data summarization

In [4]:
```
data.head(2)
```

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	NaN	NaN	4	4.5	4.5	9.65	1	0.92
1	2	312.0	107.0	4	4.0	4.5	8.87	1	0.76

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         400 non-null    int64
 1   GRE Score          399 non-null    float64
 2   TOEFL Score        399 non-null    float64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64
```


```
 2   TOEFL Score        399 non-null    float64
 3   University Rating  400 non-null    int64
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64
```


```
 8   Chance of Admit    400 non-null    float64
dtypes: float64(6), int64(3)
memory usage: 28.2 KB
```

[6]:
```
data.describe()
```

t[6]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	399.000000	399.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.619048	107.383459	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.391182	6.053848	1.143728	1.006869	0.898478	0.597325	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.167500	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.072500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

[7]:
```
data.isnull().sum()
```

t[7]:
```
Serial No.           0
GRE Score            1
TOEFL Score          1
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
```

Data Pre-processing

```
In [8]: data.drop('Serial No.', axis=1, inplace=True)
```

```
In [9]: data.rename({'Chance of Admit ': 'Chance of Admit', 'LOR ':'LOR'}, axis=1, inplace=True)
```

```
In [10]: X = data.iloc[:,:-1].values
         Y = data.iloc[:,7:].values
```

```
In [11]: print(X[:,:])
```

```
[[   nan    nan   4.   ...   4.5    9.65   1.  ]
 [312.   107.    4.   ...   4.5    8.87   1.  ]
 [316.   104.    3.   ...   3.5    8.     1.  ]
 ...
 [330.   116.    4.   ...   4.5    9.45   1.  ]
 [312.   103.    3.   ...   4.     8.78   0.  ]
 [321.   117.    4.   ...   4.     9.66   1.  ]]
```

Filling misssing values with mode

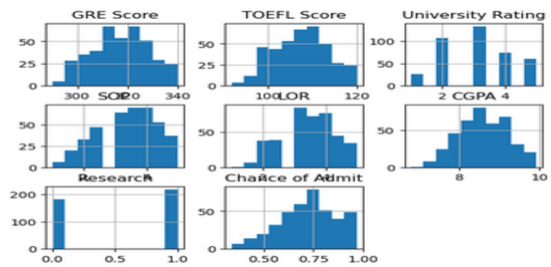
```
In [12]: from sklearn.impute import SimpleImputer
         imputer = SimpleImputer(strategy='most_frequent')
         imputer.fit(X[:, :3])
         X[:, :3]= imputer.transform(X[:,:3])
         print(X[0, :3])
```

```
[312. 110.   4.]
```

Data Visualization

```
In [13]: import matplotlib.pyplot as plt
         data.hist()
```

```
Out[13]: array([[<AxesSubplot:title={'center':'GRE Score'}>,
                 <AxesSubplot:title={'center':'TOEFL Score'}>,
                 <AxesSubplot:title={'center':'University Rating'}>],
                [<AxesSubplot:title={'center':'SOP'}>,
                 <AxesSubplot:title={'center':'LOR'}>,
                 <AxesSubplot:title={'center':'CGPA'}>],
                [<AxesSubplot:title={'center':'Research'}>,
                 <AxesSubplot:title={'center':'Chance of Admit'}>, <AxesSubplot:>]],
               dtype=object)
```

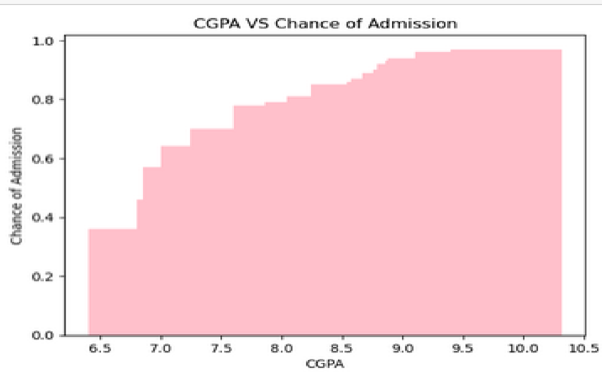



```
In [14]: GRE = pd.DataFrame(data['GRE Score'])
         GRE.describe()
```

Out[14]:

	GRE Score
count	399.000000
mean	316.619048
std	11.391182
min	290.000000
25%	308.000000
50%	317.000000
75%	325.000000
max	340.000000

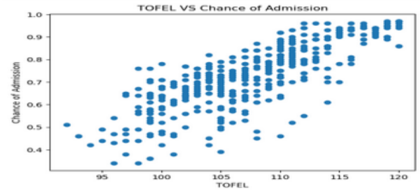
```
In [15]: plt.bar(X[:,5], Y[:,0],color = "pink")
         plt.title("CGPA VS Chance of Admission")
         plt.xlabel("CGPA")
         plt.ylabel("Chance of Admission")
         plt.show()
```



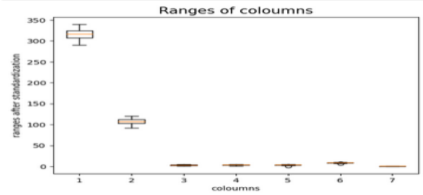

```
In [16]: print(X)
```

```
[[312.   110.    4.   ...   4.5    9.65   1.  ]
 [312.   107.    4.   ...   4.5    8.87   1.  ]
 [316.   104.    3.   ...   3.5    8.     1.  ]
 ...
 [330.   116.    4.   ...   4.5    9.45   1.  ]
 [312.   103.    3.   ...   4.     8.78   0.  ]
 [321.   117.    4.   ...   4.     9.66   1.  ]]
```

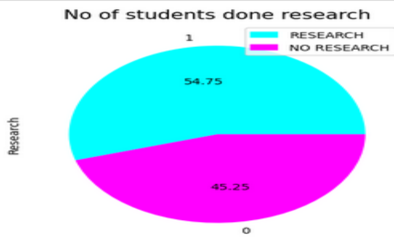
```
In [18]: plt.scatter(X[:,1], Y)
         plt.title("TOFEL VS Chance of Admission")
         plt.xlabel("TOFEL")
         plt.ylabel("Chance of Admission")
         plt.show()
```



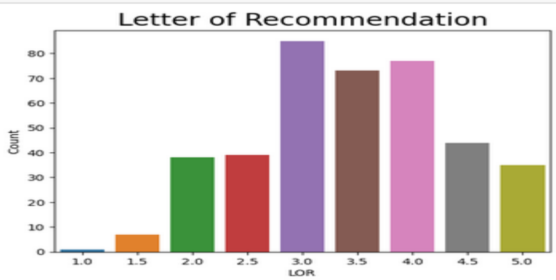

```
In [19]: plt.boxplot(X[:,:])
         plt.title('Ranges of coloumns',fontsize=15)
         plt.xlabel("coloumns")
         plt.ylabel("ranges after standardization")
         plt.show()
```




```
In [20]: data['Research'].value_counts().plot(kind='pie',textprops={'color':'black'},autopct='%.2f',cmap='cool')
         plt.title('No of students done research',fontsize=15)
         plt.legend(['RESEARCH','NO RESEARCH'])
         plt.show()
```




```
In [21]: LOR = pd.DataFrame(data.groupby(['LOR']).count()['GRE Score'])
         LOR.rename({'GRE Score':'Count'}, axis=1, inplace=True)
         sns.barplot(x = LOR.index, y = LOR['Count']).set_title('Letter of Recommendation', size='20')
         plt.show()
```

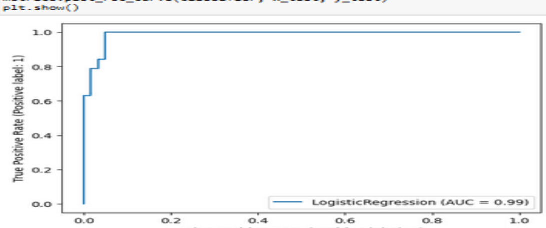


LogisticRegression

```
In [28]: from sklearn.linear_model import LogisticRegression
         classifier= LogisticRegression()
         classifier.fit(x_train, y_train)

Out[28]: LogisticRegression()

In [29]: y_logistic_pred= classifier.predict(x_test)

In [30]: print('Train Score: ', classifier.score(x_train, y_train))
         print('Test Score: ', classifier.score(x_test, y_test))

         Train Score:  0.940625
         Test Score:  0.9375

In [31]: from sklearn.metrics import confusion_matrix

         print(confusion_matrix(y_test, y_logistic_pred))

         [[60  1]
          [ 4 15]]

In [32]: from sklearn.metrics import precision_score, recall_score
         print("precision_score: ", precision_score(y_test, y_logistic_pred))
         print("recall_score: ", recall_score(y_test, y_logistic_pred))

         from sklearn.metrics import f1_score
         print("f1_score: ",f1_score(y_test, y_logistic_pred))

         precision_score:  0.9375
         recall_score:  0.7894736842105263
         f1_score:  0.8571428571428572
```

ROC curve of logistic regression

```
In [33]: from sklearn import metrics
         metrics.plot_roc_curve(classifier, x_test, y_test)
         plt.show()
```



SVC

In [34]:
```
from sklearn.svm import SVC
svm = SVC(random_state = 1)
svm.fit(x_train,y_train)
y_pred_svm = svm.predict(x_test)
print("score: ", svm.score(x_test,y_test))
```

```
score:  0.9375
```

In [35]:
```
from sklearn.metrics import precision_score, recall_score
print("precision_score: ", precision_score(y_test, y_pred_svm))
print("recall_score: ", recall_score(y_test, y_pred_svm))

from sklearn.metrics import f1_score
print("f1_score: ",f1_score(y_test, y_pred_svm))
```

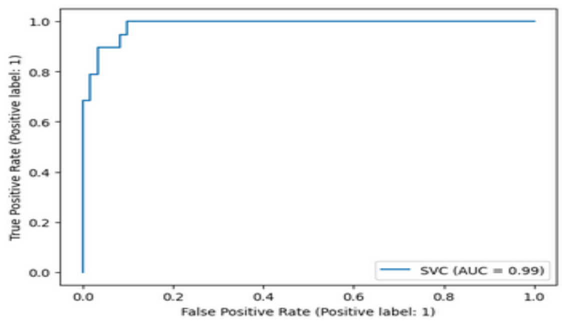
```
precision_score:  0.9375
recall_score:  0.7894736842105263
f1_score:  0.8571428571428572
```

In [36]:
```
from sklearn.metrics import confusion_matrix

print(confusion_matrix(y_test, y_pred_svm))
```

```
[[60  1]
 [ 4 15]]
```

ROC curve of SVC

In [37]:
```
from sklearn import metrics
metrics.plot_roc_curve(svm, x_test, y_test)
plt.show()
```



RandomForestClassifier

In [38]:
```
from sklearn.ensemble import RandomForestClassifier
RFC=RandomForestClassifier()
RFC.fit(x_train,y_train)
y_pred_RFC = RFC.predict(x_test)
print("score: ", RFC.score(x_test,y_test))
```

```
score:  0.925
```

In [39]:
```
from sklearn.metrics import precision_score, recall_score
print("precision_score: ", precision_score(y_test, y_pred_RFC))
print("recall_score: ", recall_score(y_test,y_pred_RFC))

from sklearn.metrics import f1_score
print("f1_score: ",f1_score(y_test, y_pred_RFC))
```

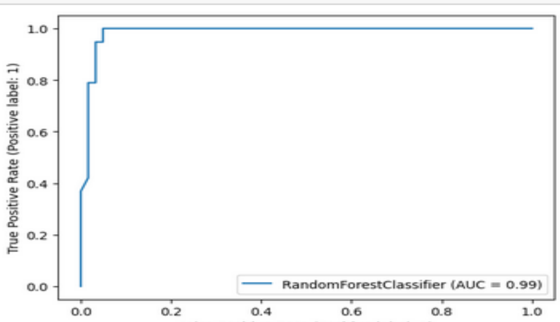
```
precision_score:  0.9333333333333333
recall_score:  0.7368421052631579
f1_score:  0.8235294117647058
```

In [40]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred_RFC))
```

```
[[60  1]
 [ 5 14]]
```

ROC curve of Random forest classifier

In [41]:
```
from sklearn import metrics
metrics.plot_roc_curve(RFC, x_test, y_test)
plt.show()
```



Naive Bayes Classifiers

In [44]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train, y_train)
y_pred_gnb = gnb.predict(x_test)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy:",metrics.accuracy_score(y_test, y_pred_gnb))
```

```
Gaussian Naive Bayes model accuracy: 0.9375
```

In [45]:
```
from sklearn.metrics import precision_score, recall_score
print("precision_score: ", precision_score(y_test, y_pred_gnb))
print("recall_score: ", recall_score(y_test,y_pred_gnb))

from sklearn.metrics import f1_score
print("f1_score: ",f1_score(y_test, y_pred_gnb))
```

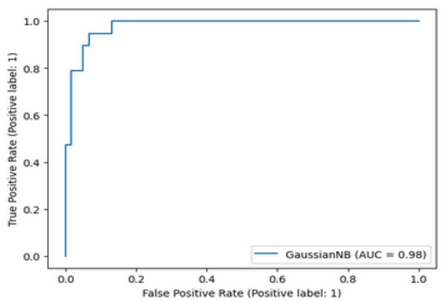
```
precision_score:  0.9375
recall_score:  0.7894736842105263
f1_score:  0.8571428571428572
```

In [46]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,  y_pred_gnb))
```

```
[[60  1]
 [ 4 15]]
```

ROC curve of Naive Bayes Classifiers

In [47]:
```
from sklearn import metrics
metrics.plot_roc_curve(gnb, x_test, y_test)
plt.show()
```



KNeighborsClassifier

In [48]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train, y_train)
y_pred_knn = knn.predict(x_test)
print("KNN model accuracy:",metrics.accuracy_score(y_test, y_pred_knn))
```

```
KNN model accuracy: 0.925
```

In [49]:
```
from sklearn.metrics import precision_score, recall_score
print("precision_score: ", precision_score(y_test, y_pred_knn))
print("recall_score: ", recall_score(y_test,y_pred_knn))
from sklearn.metrics import f1_score
print("f1_score: ",f1_score(y_test, y_pred_knn))
```

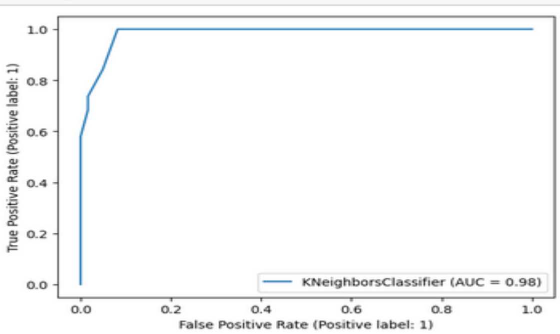
```
precision_score:  0.9333333333333333
recall_score:  0.7368421052631579
f1_score:  0.8235294117647058
```

In [50]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,  y_pred_knn))
```

```
[[60  1]
 [ 5 14]]
```

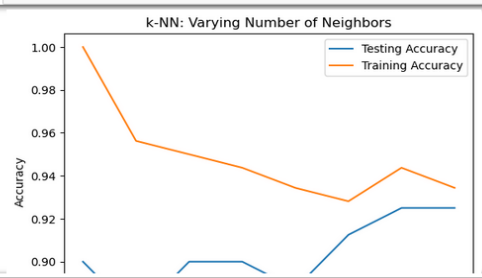
ROC curve of KNeighborsClassifier

In [51]:
```
from sklearn import metrics
metrics.plot_roc_curve(knn, x_test, y_test)
plt.show()
```



Accuracy for different values of K

In [52]:
```
no_neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(no_neighbors))
test_accuracy = np.empty(len(no_neighbors))

for i, k in enumerate(no_neighbors):
    # We instantiate the classifier
    knn = KNeighborsClassifier(n_neighbors=k)
    # Fit the classifier to the training data
    knn.fit(x_train,y_train)

    # Compute accuracy on the training set
    train_accuracy[i] = knn.score(x_train, y_train)

    # Compute accuracy on the testing set
    test_accuracy[i] = knn.score(x_test, y_test)

# Visualization of k values vs accuracy

plt.title('k-NN: Varying Number of Neighbors')
plt.plot(no_neighbors, test_accuracy, label = 'Testing Accuracy')
plt.plot(no_neighbors, train_accuracy, label = 'Training Accuracy')
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.show()
```



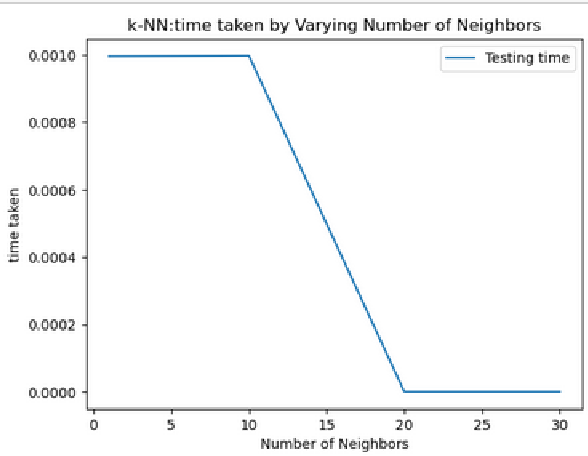
k-NN:time taken by Varying Number of K

Decision tree using cart model

In [54]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(x_train,y_train)
y_pred_clf = clf.predict(x_test)
print("score: ", clf.score(x_test,y_test))
```

```
score:  0.8875
```

In [53]:
```
no_neighbors = [1,10,20,30]
time_taken = np.empty(len(no_neighbors))

import time
for i, k in enumerate(no_neighbors):
    start=time.time()
    # We instantiate the classifier
    knn = KNeighborsClassifier(n_neighbors=k)
    # Fit the classifier to the training data
    knn.fit(x_train,y_train)
    end=time.time()
    time_taken[i]=end-start

# Visualization of k values vs accuracy

plt.title('k-NN:time taken by Varying Number of Neighbors')
plt.plot(no_neighbors, time_taken, label = 'Testing time')
plt.legend()
plt.xlabel('Number of Neighbors')
plt.ylabel('time taken')
plt.show()
```



In [55]:
```
from sklearn.metrics import precision_score, recall_score
print("precision_score: ", precision_score(y_test, y_pred_clf))
print("recall_score: ", recall_score(y_test,y_pred_clf))

from sklearn.metrics import f1_score
print("f1_score: ",f1_score(y_test, y_pred_clf))
```

```
precision_score:  0.7777777777777778
recall_score:  0.7368421052631579
f1_score:  0.7567567567567567
```

CROSS VALIDATION SCORES

In [56]:
```
from sklearn.model_selection import cross_val_score
```

In [57]:
```
cross_val_score(LogisticRegression(),x_train,y_train)#average=0.93125
```

Out[57]:
```
array([0.921875, 0.890625, 0.984375, 0.921875, 0.9375  ])
```

In [58]:
```
cross_val_score(SVC(),x_train,y_train)#average=0.928125
```

Out[58]:
```
array([0.921875, 0.890625, 0.984375, 0.921875, 0.921875])
```

In [59]:
```
cross_val_score(RandomForestClassifier(),x_train,y_train)#average=0.946875
```

Out[59]:
```
array([0.921875, 0.90625 , 1.      , 0.9375  , 0.96875 ])
```

In [60]:
```
cross_val_score(GaussianNB(),x_train,y_train)#average=
```

Out[60]:
```
array([0.921875, 0.890625, 0.9375  , 0.875   , 0.9375  ])
```

In [61]:
```
cross_val_score(KNeighborsClassifier(),x_train,y_train)#average=
```

Out[61]:
```
array([0.9375  , 0.921875, 0.96875 , 0.921875, 0.90625 ])
```

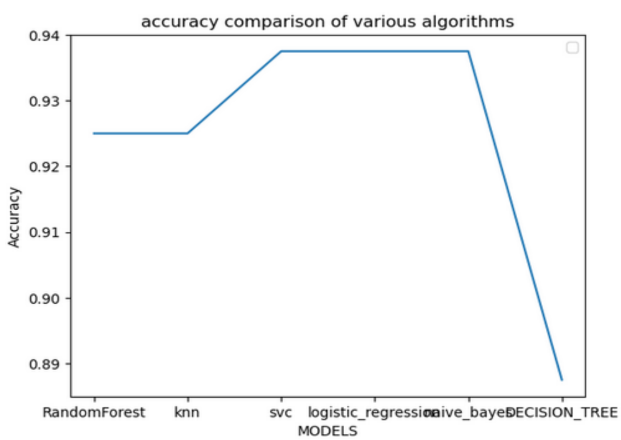
In [62]:
```
cross_val_score(DecisionTreeClassifier(),x_train,y_train)#average=
```

Out[62]:
```
array([0.9375  , 0.890625, 0.9375  , 0.90625 , 0.84375 ])
```

accuracy comparison of various algorithms

In [63]:
```
models=['RandomForest','knn','svc','logistic_regression','naive_bayes','DECISION_TREE']
accuracy=[metrics.accuracy_score(y_test, y_pred_RFC),metrics.accuracy_score(y_test, y_pred_knn),metrics.accuracy_score

plt.title('accuracy comparison of various algorithms')
plt.plot(models, accuracy)
plt.legend()
plt.xlabel('MODELS')
plt.ylabel('Accuracy')
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored wh
en legend() is called with no argument.
```



RESULT AND DISCUSSION

- This project is successfully completed with the best model possible.
- Random forest is the best from all the models
- The result of the comparative study is shown below:

```
from sklearn.ensemble import RandomForestClassifier
RFC=RandomForestClassifier()
RFC.fit(x_train,y_train)
y_pred_RFC = RFC.predict(x_test)
print("score: ", RFC.score(x_test,y_test))
```

```
score:  0.925
```

```
from sklearn.metrics import precision_score, recall_score
print("precision_score: ", precision_score(y_test, y_pred_RFC))
print("recall_score: ", recall_score(y_test,y_pred_RFC))

from sklearn.metrics import f1_score
print("f1_score: ",f1_score(y_test, y_pred_RFC))
```

```
precision_score:  0.9333333333333333
recall_score:  0.7368421052631579
f1_score:  0.8235294117647058
```

REFERENCES

- "Prediction fora University Admission Using Machine Learning" by Chitra Apoorva D.A, Malepati Chandu Nath, Peta Rohith, Swaroop S, Bindushree S - Blue Eyes Intelligence Engineering & Sciences Publication. International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8, Issue-6 March 2020
- "Machine Learning Basics with the K-Nearest Neighbors Algorithm"- https://towardsdatascience.com/machine-learningbasics-with-the-k-nearest-neighbors-algorithm
- "Random Forest Regression"
  - https://www.kaggle.com/dansbecker/random-forests
- "Data pre-processing & Machine Learning"https://archieve.ics.uci.edu/ml/index.php
- "User Interface Design" - https://pidoco.com/en/help/ux/user-interface-design
- "Graduate Admission Prediction Using Machine Learning" by Sara Aljasmi, Ali Bou Nassif, Ismail Shahin, Ashraf Elnagar - ResearchGate Publication, December 2020
- Sujay S "Supervised Machine Learning Modelling & Analysis For Graduate Admission Prediction" Published in International Journal of Trend in Research and Development (IJTRD), ISSN: 2394-9333, Volume-7 | Issue-4 , August 2020

▶