

**R & D Project Report**

**Academic Year- 2021-22**

On

Analysis of academic performance, IP and job profile

Submitted by:

|  |  |  |
| --- | --- | --- |
| Gonuguntla Harichandana | BT19GCS038 | CSE |
| Siddharth Mehta | BT19GCS113 | CSE |
| Utkarsh Nagar | BT19GCS080 | CSE |

**Supervisors**

Prof. Ratna Sanyal

Prof. Eswaran Narasimhan

Ms. Geetica Rastogi

Logo, company name

Description automatically generated

=================================================================

**CERTIFICATE BY SUPERVISOR(S)**

This is to certify that the present R&D project entitled Analysis of academic performance, IP and job profile being submitted to NIIT University, Neemrana, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology, in the area of CSE, embodies faithful record of original research carried out by Utkarsh Nagar, Siddharth Mehta and Gonuguntla Harichandana. They have worked under our guidance and supervision and that this work has not been submitted, in part or full, for any other degree or diploma of NIIT or any other University.

Place: Name of the Supervisor(s) with signature

Date:

Logo, company name

Description automatically generated

=================================================================

**DECLARATION BY STUDENTS**

We hereby declare that the project report entitled Analysis of academic performance, IP and job profile which is being submitted for the partial fulfilment of the Degree of Bachelor of Technology, at NIIT University, Neemrana, is an authentic record of our original work under the guidance of Prof. Ratna Sanyal, Prof. Eswaran Narasimhan and Ms. Geetica Rastogi. Due acknowledgements have been given in the project report to all other related work used. This has previously not formed the basis for the award of any degree, diploma, associate/fellowship or any other similar title or recognition in NIIT University or elsewhere.

Place:

Date:

|  |  |  |
| --- | --- | --- |
| Gonuguntla Harichandana | BT19GCS038 | CSE |
| Siddharth Mehta | BT19GCS113 | CSE |
| Utkarsh Nagar | BT19GCS080 | CSE |

**Table of contents**

1. Introduction…………………………………………………. 05
2. Problem Statement…………………………………………. .06
3. Literature review……………………………………………. 06
4. Proposed methodology
   1. Workflow……………………………………….. .08
      1. Data Collection
      2. OneHote Encoding
      3. Applying the Algorithm and Training and testing
   2. Technology……………………………………… 10
      1. Decision Tree
      2. SVM
      3. XGBoost
5. Result ……………………………………………………….. 13
6. Comparison with other existing work and Analysis ………. 14
7. Concluding Remarks ……………………………………….. 15
8. Future Scope………………………………………………… 15
9. References……………………………………………………16
10. Appendix……………………………………………………..17
11. Source Code………………………………………………… 18

**1 Introduction**

Increasing competition in today's world becomes very burdensome so students must plan and arrange themselves from the start of their studies.

Recruiters also assess prospects based on a variety of characteristics before deciding whether or not to hire them, as well as determining the best job and career path for them if they are hired. There are many other sorts of positions, such as Software Engineer, Software Developer, and Machine Learning Engineer. Some prior experience is necessary to be placed in any of these occupations. As a consequence, recruiters assess the candidate's abilities, capabilities, and interests and place them in the ideal job for them. As a result, when suggestions are generated based on the inputs, the prediction system would make their recruiting job a lot easier. Traditional programming and algorithms are incapable of providing the best possible output classification and prediction due to the large number of input parameters and final output classes. As a result, powerful machine learning methods such as SVM, OneHot encoding, and XGBoost are employed.

Machine Learning techniques enable the solution of extremely complex tasks and issues with minimal human intervention. This research focuses on classification and forecasting. Let's start with a definition of classification and prediction. supervised and unsupervised learning can be used to solve the bulk of machine learning problems. It is alluded to in the event that the last class names are known early and every one of extra information things are to be allocated one of the accessible class marks. When the final output classes and sets are unknown, unsupervised learning is used to identify similarities between data points and their characteristics before categorising them based on these similarities. Classification is classified as supervised. Input parameters are provided, and a predefined class label is assigned depending on their qualities. There are additional options, such as grouping and regression. The suitable model is selected based on the problem kind.

Methods such as OneHot encoding and Decision tree are utilized here. After training and evaluating the data with these, we use the algorithm that produces the best accurate results for our subsequent processing. So, the first step is to forecast the output using all of the techniques provided above, then analyze the findings and proceed with the most accurate approach. Finally, in this project we are going to use advanced Machine Learning techniques to train our model in order to predict the job profile of a student with better accuracy.

**2 Problem Statement**

The problem statement given to us is to “Analyze the data containing marks scored in all the semesters by a student and predict his/her IP and Job profile. Because competition is growing at an exponential rate currently.

It's vital to evaluate students' performance on a regular basis, monitor their progress with their interests, marks and goals. This aids individuals in growing and encourages them to select a better professional path if their talents are insufficient to meet their objectives, as well as evaluation of their performance beforehand. Also recruiters assess applicants on a range of criteria before selecting whether or not to consider them for the post and, if hired, establishing the best employment and career route for them. The fundamental purpose of this project is to forecast the employment profiles of the next batch of students based on an analysis of their grades in 10th grade, 10+2, all semesters of college, special, and prior batches from the same institutions.

**3 Literature review**

One **[1]** of the research papers discussed the link between students' academic profiles before admission and their final academic performance. A data sample of students from one of the Federal Polytechnics in Nigeria's southwest was used. The grade point average of the student is used to define academic performance (GPA). This study focused on developing a model for forecasting student performance based on 'O' level scores and their first three semesters at each semester using data mining techniques. The findings of rusticated and expelled students were removed by data cleaning. When comparing SVM to other approaches such as KNN, linear regression and Decision trees the results reveal that SVM outperforms all others. The SVM algorithm (kernel) parameters were also tweaked to increase accuracy, and the results suggest that the RBF kernel with the penalty (C=100) works best. RBF and SVM provided the greatest training performance of 94 per cent and 97 per cent forecasting accuracy, respectively, outperforming other cutting-edge ML techniques such as decision trees,KNN, and others.

Another research paper **[2]** on XGBoost academic forecasting and analysis modelling talked about the analysis and evaluation of student success as an important aspect of both teaching and school routine administration. Scientifically analysing and assessing students' academic achievements allows instructors to not only correctly comprehend the students' learning status, but also allow students to understand their own learning conditions and give required analysis for instructional management and improvement. This work employs the XGBoost algorithm to categorise and assess students' performance based on a statistical analysis of fundamental data in order to evaluate students' learning situations thoroughly, objectively, and appropriately. The student's performance data is statistically gathered according to statistical knowledge for curriculum relevance.

Another research paper **[3]** presents a review of the research on student performance utilising decision trees and various decision tree algorithms in the educational area. The success of any educational institution is determined by the performance of its pupils. Prediction and analysis of student performance are critical factors in institute improvement. Attendance, internal marks, final grades, and other factors all have an impact on student achievement. This form of prediction assists instructors and institutions in identifying poor pupils in order to help them improve their grades and make a range of rapid selections in any circumstance. For performance analysis in data mining, several approaches such as classification, regression, association, and clustering are utilised. These approaches are known as knowledge discovery databases in the educational sphere. This is used to detect the most recent hidden trends in student databases for performance analysis.

The research paper **[4]** based on Fuzzy Logic principles, suggested (NFES) for evaluating student academic achievement. It presents the basics of fuzzy logic and shows how educators might use these ideas to evaluate student academic achievement. The New Fuzzy Expert System (NFES) suggested in this study intends to adaptively change training for each individual learner based on his/her personal learning rate. This implies that the NFES will be able to monitor the student's development and make decisions regarding the next steps in training. Several approaches based on fuzzy logic techniques have been presented in order to give a realistic mechanism for evaluating student academic achievement and comparing the findings (performance) to current statistical methods.

Another research paper **[5]** talked about how decision trees are a prominent way for modelling classifiers. Researchers from a variety of domains, including statistics, machine learning, pattern recognition, and data mining, have addressed the topic of constructing a decision tree from provided data. Unified algorithmic framework is used for extracting these methods and covers various splitting criteria and trimming procedures.

**4 Proposed methodology:**

**4.1 Data Collection**

Data collected in a variety of methods will be disorganised, with erroneous data values, null values, and undesired information. Data collection involves

* Cleaning of data
* Replacement with acceptable data
* Elimination of null data
* Exporting the data into the expected format

In our case, two types of datasets were involved, one being provided by the University. That data comprises 10th and 12th marks of students with their semester wise cgpa and percentage obtained in the semesters. The other dataset includes around 10k records containing knowledge of student hobbies, interest, specialization, achievements etc.

**4.2 OneHot Encoding**

OneHot Encoding is a method for turning categorical values in collected data into numerical or ordinal representations so that computers can interpret them and improve prediction results using learning algorithms. Simply said, OneHot encoding transforms categorical data into a format that is compatible with a variety of machine learning methods. Almost all machines are compatible with this algorithm. Only a few algorithms, such as random forest, are capable of handling category values well. OneHot encoding isn't required in those situations.

Although OneHot encoding looks to be a difficult approach, it is handled by most modern machine learning algorithms. The procedure is simple: if a data collection contains yes and no values, the integer encoder gives them the numbers 1 and 0. As long as the values for yes and no stays at 1 and 0 this strategy can be used. This is known as integer encoding since these fixed numbers are assigned or allocated to these specified labels. However, consistency is critical in this scenario since we should be able to reliably obtain the labels if we invert the encoding later, which is especially important in the case of prediction.

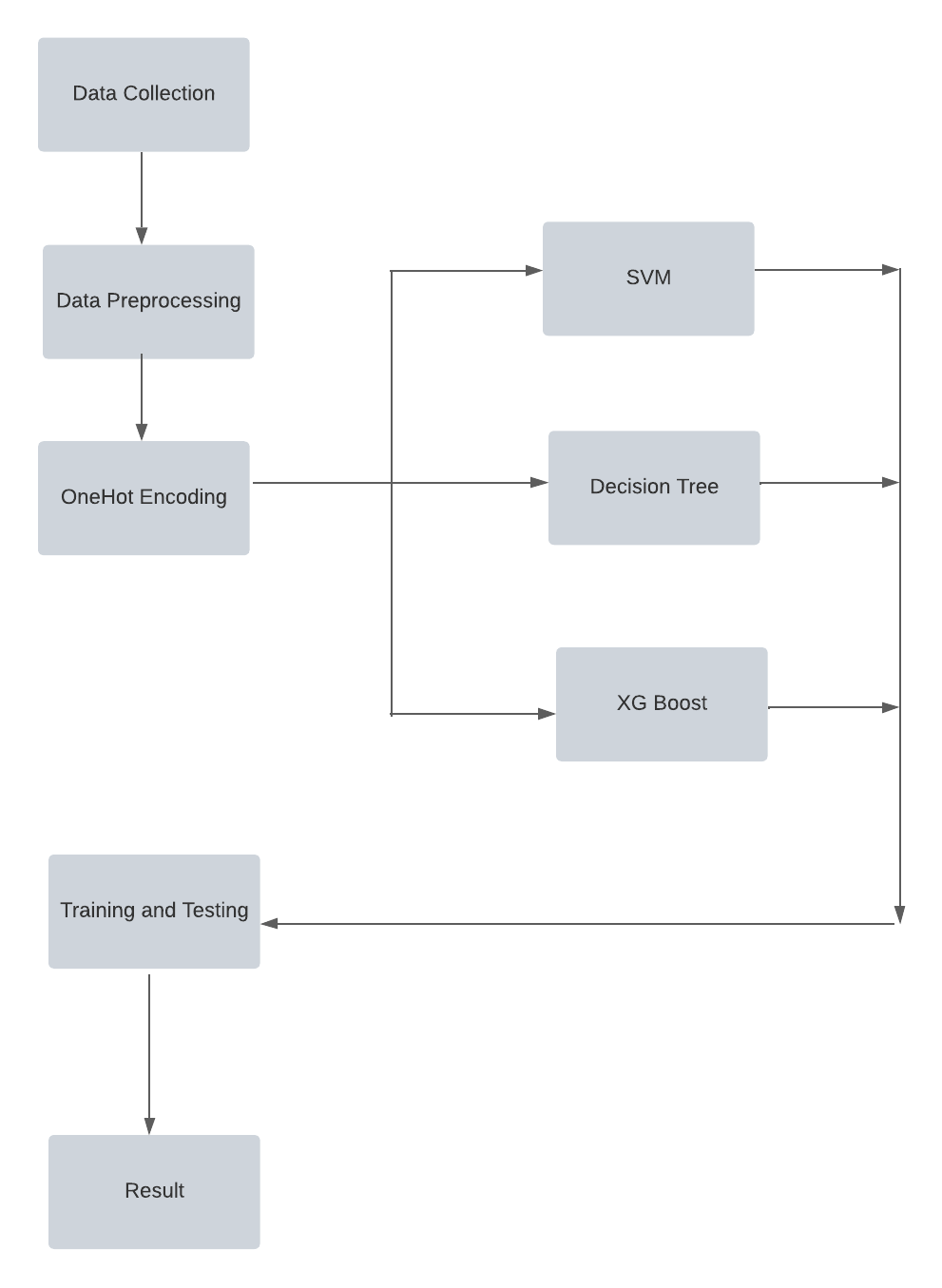
****

Fig. 1: Workflow

**4.3 Applying the Algorithm and Training and testing**

The next stage is to apply algorithms to data and note and observe the outcomes. The algorithms are used to increase accuracy at each level of the fashion mentioned in the diagram.

After data processing and training, the final stage is testing. The vast majority of the data set is used for training, with the remaining 20% used for testing. Training is the process of teaching a computer to learn and make subsequent predictions.

Whereas testing is having a data set with already labelled output and testing the model to see if it is performing properly and making the correct prediction.

**5 Technology**

The project revolves on machine algorithms and approaches **[11]**. It's easier to write code in Python, which comes with a large number of built-in libraries that make writing a machine learning application simpler. (In addition, the data set contains 25% randomly produced data that does not follow a pattern and was generated using Java.). The data processing is described below:

To normalise data, we used Pandas to analyse the raw data, which was in excel format and then used data frames and NumPy to access the data at various points of the file, i.e., in tabular format, making it compatible to fit into the algorithm. Sklearn, the most well-known machine learning package, was used to import many tools and approaches, such as data decomposition to a reduced dimensional space. OneHotEncoder to encode categorical characteristics as a one-hot numeric array, LabelEncoder to standardise the data labels to have unit norm, use a normalizer to scale the input data set on a scale of 0 to 1. Accuracy score to compute the accuracy count of accurate prediction, SVM to handle classification or regression issues, and XGBClassifier to create quick and high-performance gradient boosting tree models.

**Dataset:** The institution has contributed raw data, which is used to analyse the data to detect and forecast trends.

**Algorithms:** In this study, we concentrated on Support Vector Machine (SVM), XGBoost, and Decision Tree algorithms, where the data is processed by these algorithms and the result is used for the study.

**5.1 SVM**

SVM is a supervised learning system that may be used to detect outliers, categorise data, and conduct regression **[13]**. SVM entails determining the best line or decision boundary for dividing n-dimensional space into classes in the future, so that fresh data points may be quickly assigned to the right category. The selection of extreme points/vectors using SVM, contributes to the formation of the decision boundary.

In the model with the university’s dataset, SVM is the algorithm that produced the result with maximum accuracy.

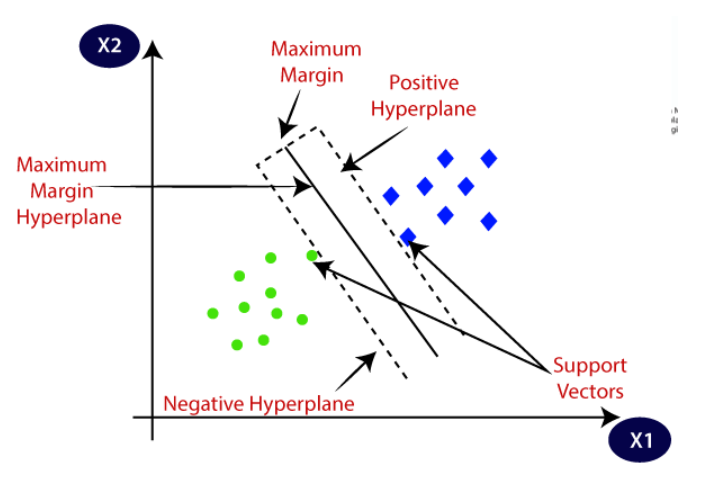


Fig. 2: classification using a hyperplane

(<https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>)

Algorithmic steps:

1. Preprocessing of Data
2. Applying SVM classification with dataset
3. Result Prediction
4. Creating the confusion matrix
5. Set result visualization

**5.2 XGBoost**

Gradient Boosting is primarily concerned with model performance and calculation time. It drastically cuts down on time and improves the model's performance. The technique's implementation offers the following advantages: automatic missing value management, a block structure for concurrent tree construction, and ongoing training to support a trained model with new data. Final prediction involves a combination of new models, mistakes of previous models, and learning from them which will be used to infer new knowledge. They help with categorization issues like the one we're dealing with right now **[6]**.

**5.3 Decision Based Tree**

A decision tree is a form of flow chart that is used to represent the decision-making process by outlining several courses of action and their potential results. It belongs to the category of supervised learning algorithms.

Tree-based learning algorithms are one of the most successful and widely used supervised learning approaches. Tree-based prediction models provide a high level of accuracy, stability, and interpretability.

Unlike linear models, they successfully map nonlinear interactions. They are adaptable in their approach to resolving any issue. The XGBoost algorithm is an upgraded version of this large decision tree **[7]**.

**Working of algorithm:**

1. Asusual, we’ll start the tree with a root node containing a complete dataset.
2. Finding the best attribute in dataset
3. Subdivision based on best attribute, into subsets
4. Create a new node in decision tree with the selected attribute
5. Use recursion to and step 3 and repeat until the termination condition is satisfied

**6 Result**

In the vast majority of circumstances, SVM proves to be more precise than other algorithms in terms of outcomes, and SVM is better suited to data which is not evenly distributed. The advantage of SVM is that it includes a technique called kernel, which may be used to tackle any difficult problem. In addition, SVM is very rapid in execution, which implies it requires less time than other artificial neural networks. And adding or updating the values to a very little to the data set does not really affect the results alongside proving that the SVM has faster prediction and better accuracy and time complexity of O(N2K) where k is the number of support vectors.

Decision trees, unlike SVM, are not resistant to changes in the data set; in most circumstances, a little change in the data set can result in a huge change in the structure of the best decision tree. Additionally, the decision tree algorithm takes longer to train the model with more time complexity, making it more expensive, but it does not require data to be scaled or normalised. Meanwhile, XGBoost is hardly scalable and it is very sensitive to outliers.

Also, XGBoost can predict the future at a good level and is very good at applying real world business problems. XGBoost does not consume as much time as other algorithms making it easier to use **[8]**.

Another reason an algorithm performs better than others in comparison is because the data set used to train the model may contain more negative samples than positive samples, or that a large number of unfavourable situations may outnumber favourable examples. As a result, we might acquire great precision to a certain algorithm in some instances.

Based on the above values and the analysis made based on the data(university provided) fed to the models, SVM can be recommended much compared to others for good predictions on any data.

**7 Comparison with other existing work and Analysis:**

Upon referring to the research papers **[9][10][11]**, based on the data that they have fed the models, the analysis of different algorithms comes by:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Our Model’s Accuracy** | |
| **Technique** | **Accuracy** | **Generated Dataset** | **1-year College Dataset** |
| Decision Tree | 56.25% - 98.86% | 29.00% | 14.70% |
| KNN | 69.00% - 83.00% | - | - |
| Naive Bayes | 50.00% - 91.57% | - | - |
| SVM | 80.00% - 98.00% | 56.00% | 15.29% |
| XGBoost | 73.00% - 94.00% | 57.50% | 11.76% |

(\*Values might change to an extent based on the data fed to the models)**[12]**

The main focus of this study includes the prediction of results based on past data and suggest relative job profiles to the user alongside, comparison of the algorithms based on this application brings out the working of the algorithms and their accuracy. Most of the research papers we referred to are focussing on single algorithms and different real time data sets according to their availability which will result in different predictions.

Our study mainly focuses on how the predictions are changing and the accuracy varies on the same data but different algorithms and thereby concluding the best-case algorithm which suits the data we feed to the model. And there is minimal research on career prediction, so this study would add another contribution to this genre. As mentioned in the other section of this study, the data set contains a quarter of data set information randomized i.e., false data and being fed to the models in order to train the data with the possible and feasible scenarios to learn unlike other research papers.

**8 Concluding Remarks**

It is feasible to build a sophisticated online application that obtains student parameters based on awareness, interests, and abilities. As a result, the system will be more precise, and the results will be more precise.

Furthermore, decision trees have a number of drawbacks, including overfitting, no pruning, and a lack of capacity to cope with null and missing values, while a few methods struggle with large amounts of data set of values All of this may be taken into account, making it even more dependable.

All of this may be taken into account, and even more dependable, accurate, and sophisticated algorithms can be employed. The project will be more accurate as a consequence, and the user will be able to depend on the findings.

**9 Future Scope**

We are now analysing student grades and utilizing decision trees to forecast their employment profiles. In the future, we may compare the output of the present model to the output of SVM and SG Boost. Also, a web application with a simple user interface may be constructed so that people can input student data and the student's employment profile would be predicted. The model may be improved by including other elements such as abilities and extracurricular activities undertaken by pupils.

We can also train previous year data of our university which can be later used by placement cell (CIC) during the time of placements. This can help both students and university as they can easily monitor their progress with prediction.

We can also implement mobile applications with this functionality so that every student can use it, this application will help them portability use it.

Recruiters also analyse candidates on several aspects when hiring new employees and make a final decision whether to hire them or not, and if hired, discover the appropriate job and career area for them. As a result, we can utilize technology to make this process easier, allowing recruiters to examine these abilities, talents, and interests and position the prospect in the best employment for them. These kinds of activities may be made relatively simple since recommendations are generated depending on the inputs.

**10 References**

[1] R.S Yadav, A.K Soni, S Pal, A Study Of Academic Performance Evaluation Using Fuzzy Logic Techniques, 2014 International Conference on “Computing for Sustainable Global Development”, 54-59.

[2] E Wakelam, A Jefferies, A Neil Davey, Yi Sun, The Potential for Using Artificial Intelligence Techniques to Improve e-Learning Systems, ECEL 2015 Conference proceedings, October 2015.

Terence Shin, All Machine Learning Models Explained in 6 Minutes, Jan 6, 2020.

[3] S Harinath, A Prasad, Suma H.S, Suraksha A, T.Mathew, Student Placement Prediction Using Machine Learning, International Research Journal of Engineering and Technology, April 2019, 4577-4579

[4] P.KaviPriya, “A Review on Predicting Students’ Academic Performance Earlier, Using Data Mining Techniques”, International Jour-nal of Advanced Research in Computer Science and Software Engineering

[5] Ali Daud, Naif Radi Aljohani, “Predicting Student Performance using Advanced Learning Analytics”, 2017 International World Wide Web Conference Committee (IW3C2).

[6] Marium-E-Jannat,SaymaSultana,Munira Akther, “A Probabilistic Machine Learning Approach for Eligible Candidate Selection”, International Journal of Computer Applications (0975 – 8887)Volume 144 – No.10, June 2016

[7] Sudheep Elayidom, Dr. Sumam Mary Idikkula, “Applying Data mining using Statistical Techniques for Career Selection”, International Journal of Recent Trends in Engineering, Vol. 1, No. 1, May 2009.

[8] Mukesh Kumar, Yass Khudheir Salal, Systematic Review of Predicting Student's Performance in Academics, International Journal of Engineering and Advanced Technology , February 2019, 54-61.

[9] S.A. Oloruntoba1 ,J.L.Akinode, Student Academic Performance Prediction using Support Vector Machine, International journal of Engineering Sciences & Research Technology, December 2017, 588-598.

[10] Ting Hu, Ting Song, Research on XGboost academic forecasting and analysis modelling, The Second International Conference on Physics, Mathematics and Statistics 22–24 May 2019.

[11] Mrinal Pandey, Vivek Kumar Sharma, A Decision Tree Algorithm Pertaining to the Student Performance Analysis and Prediction, International Journal of Computer Applications, January 2013.

[12] Lior Rokach, Oded Maimon, Decision Trees, January 2005, 166-192.

**Appendix**

|  |  |
| --- | --- |
| **Term** | **Description** |
| SVM | Support Vector Machine (SVM) creates the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point into the correct category in the future. |
| .CSV file format | .CSV (comma-separated values) is a basic file format for storing table data in software that stores data in tables, such as Microsoft Excel(.xls,xlsx) or OpenOffice Calc. |
| Fuzzy logic | Fuzzy logic is used to deal with the concept of partial truth, where the truth value can be somewhere between true and false]. |
| KNN | KNN algorithm is a supervised machine learning model. It predicts a target variable using one or multiple independent variables. |
| OneHot Encoding | One-hot Encoding is a vector representation in which all of the elements in a vector are 0 except for one, which has a value of 1, representing a boolean identifying the element's category. |
| NFES | a New Fuzzy Expert System |
| Fig 1: Workflow | The figure shows the methodology and workflow adopted during the course of the project. |
| Fig 2 | The figure shows two different categories that are classified using a decision boundary or hyperplane |

**11 Source Code**

**11.1 Project Code for generated data :**

|  |
| --- |
| **#--------------------Importing Required Libraries/Modules-----------------------#**  import pandas as pd  import numpy as np  from sklearn import decomposition  import matplotlib.pyplot as plt  **#--------reading the .csv file as input------------#**  dataset = pd.read\_csv("rnd\_data.csv")  **#---------Testing by displaying whether data is loaded properly or not-----------#**  data = dataset.iloc[:,:-1].values  label = dataset.iloc[:,-1].values  len(data[0])  dataset.iloc[:]  **#---------------Applying OneHot & Label Encoding-----------#**  from sklearn.preprocessing import LabelEncoder, OneHotEncoder  labelencoder = LabelEncoder()  xa=0.813664  **#---------------conversion of all categorical column values to vector/numerical--------#**  for i in range(16):  data[:,i] = labelencoder.fit\_transform(data[:,i])  data[:]  **#--------------normalizing the non-categorical column values---------#**  data1=data[:]  normalized\_data = Normalizer().fit\_transform(data1)  print(normalized\_data.shape)  da=0.8383  normalized\_data  data2=data[:]  data2.shape  df1 = np.append(normalized\_data,data2,axis=1)  sa=0.8516  df1.shape  **#--------------------------Adding Headers-----------------------#**  X1 = pd.DataFrame(df1,columns=['Specialization','10th','10+2','CGPA at the End of Semester I','CGPA at the End of Semester II','CGPA at the End of Summer Term I','CGPA at the End of Semester III','CGPA at the End of Semester IV','CGPA at the End of Summer Term II','CGPA at the End of Semester V','CGPA at the End of Semester VI','CGPA at the End of Semester VII','CGPA at the End of Semester VIII','CGPA at the End of Summer Term IV','CGPA at the End of Semester IX','IP Profile','Specialization','10th','10+2','CGPA at the End of Semester I','CGPA at the End of Semester II','CGPA at the End of Summer Term I','CGPA at the End of Semester III','CGPA at the End of Semester IV','CGPA at the End of Summer Term II','CGPA at the End of Semester V','CGPA at the End of Semester VI','CGPA at the End of Semester VII','CGPA at the End of Semester VIII','CGPA at the End of Summer Term IV','CGPA at the End of Semester IX','IP Profile'])  X1.head()  **#------------------Encoding Final Output column Values------------#**  label = labelencoder.fit\_transform(label)  print(len(label))  y=pd.DataFrame(label,columns=["Suggested Job Role"])  y.head()  **#------------------Training and testing with Decision Tree----------------#**  **#------importing modules---------------#**  from sklearn import tree  from sklearn.model\_selection import train\_test\_split  from sklearn import preprocessing  from sklearn.metrics import accuracy\_score  **#---------------specifying percentage of test data from whole data--------#**  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X1,y,test\_size=0.2,random\_state=10)  **#-----------------classifying with decision tree-----------------------#**  clf = tree.DecisionTreeClassifier()  clf = clf.fit(X\_train, y\_train)  from sklearn.metrics import confusion\_matrix,accuracy\_score  y\_pred = clf.predict(X\_test)  Y\_pred  **#----------------calculating confusion vector values matrix and accuracy-------------#**  cm = confusion\_matrix(y\_test,y\_pred)  accuracy = accuracy\_score(y\_test,y\_pred)  print("confusion matrics=",cm)  print(" ")  print("accuracy=",accuracy\*100)  **#--------performing decision tree classification with entropy as measure------------#**  clf\_entropy = tree.DecisionTreeClassifier(criterion = "entropy", random\_state = 10)  clf\_entropy.fit(X\_train, y\_train)  entropy\_y\_pred=clf\_entropy.predict(X\_test)  cm\_entopy = confusion\_matrix(y\_test,entropy\_y\_pred)  entropy\_accuracy = accuracy\_score(y\_test,entropy\_y\_pred)  print("confusion matrix=",cm\_entopy)  print(" ")  print("accuracy=",entropy\_accuracy\*100)  **#------classification with svm------------------------#**  from sklearn import svm  clf = svm.SVC()  clf.fit(X\_train, y\_train) #------giving test data as input----#  svm\_y\_pred = clf.predict(X\_test) #--------doing prediction-----#  **#-----------calculating confusion matrix and accuracy----------#**  svm\_cm = confusion\_matrix(y\_test,svm\_y\_pred)  svm\_accuracy = accuracy\_score(y\_test,svm\_y\_pred)  print("confusion matrix=",svm\_cm)  print(" ")  print("accuracy=",svm\_accuracy\*100)  **#--------------classification using xgboost--------------#**  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X1,y,test\_size=0.3,random\_state=10)  X\_train.shape  **#------------converting values of training and testing data into int64 datatype-------#**  X\_train=pd.to\_numeric(X\_train.values.flatten())  X\_train=X\_train.reshape((118,32))  **#-------------importing and defining xgboost functions-----#**  from xgboost import XGBClassifier  model = XGBClassifier()  **#-----------training and testing with xg boost------#**  model.fit(X\_train, y\_train)  xgb\_y\_pred = clf.predict(X\_test)  **#-----calculating confusion matrix and accuracy after boosting--------#**  xgb\_cm = confusion\_matrix(y\_test,xgb\_y\_pred)  xgb\_accuracy = accuracy\_score(y\_test,xgb\_y\_pred)  print("confusion matrix=",xgb\_cm)  print(" ")  print("accuracy=",xgb\_accuracy\*100) |

**11.2 University’s Data**

|  |
| --- |
| **#--------------------Importing Required Libraries/Modules-----------------------#**  import pandas as pd  import numpy as np  from sklearn import decomposition  import matplotlib.pyplot as plt  **#--------reading the .csv file as input------------#**  dataset = pd.read\_csv("roo\_data.csv")  **#---------Testing by displaying whether data is loaded properly or not-----------#**  data = dataset.iloc[:,:-1].values  label = dataset.iloc[:,-1].values  len(data[0])  Dataset.iloc[:,14:38]  Dataset.iloc[:,14:38]  **#---------------Applying OneHot & Label Encoding-----------#**  from sklearn.preprocessing import LabelEncoder, OneHotEncoder  labelencoder = LabelEncoder()  xa=0.813664  **#---------------conversion of all categorical column values to vector/numerical--------#**  for i in range(14,38):  data[:,i] = labelencoder.fit\_transform(data[:,i])  data[:5]  Data[:5,14:]  **#--------------normalizing the non-categorical column values---------#**  from sklearn.preprocessing import Normalizer  data1=data[:,:14]  normalized\_data = Normalizer().fit\_transform(data1)  print(normalized\_data.shape)  da=0.8383  normalized\_data  data2=data[:,14:]  data2.shape  df1 = np.append(normalized\_data,data2,axis=1)  sa=0.8516  df1.shape  **#--------------------------Adding Headers-----------------------#**  X1 = pd.DataFrame(df1,columns=['Academic percentage in Operating Systems', 'percentage in Algorithms',  'Percentage in Programming Concepts',  'Percentage in Software Engineering', 'Percentage in Computer Networks',  'Percentage in Electronics Subjects',  'Percentage in Computer Architecture', 'Percentage in Mathematics',  'Percentage in Communication skills', 'Hours working per day',  'Logical quotient rating', 'hackathons', 'coding skills rating',  'public speaking points', 'can work long time before system?',  'self-learning capability?', 'Extra-courses did', 'certifications',  'workshops', 'talent tests taken?', 'olympiads',  'reading and writing skills', 'memory capability score',  'Interested subjects', 'interested career area ', 'Job/Higher Studies?',  'Type of company want to settle in?',  'Taken inputs from seniors or elders', 'interested in games',  'Interested Type of Books', 'Salary Range Expected',  'In a Relationship?', 'Gentle or Tuff behaviour?',  'Management or Technical', 'Salary/work', 'hard/smart worker',  'worked in teams ever?', 'Introvert'])  X1.head()  **#------------------Encoding Final Output column Values------------#**  label = labelencoder.fit\_transform(label)  print(len(label))  y=pd.DataFrame(label,columns=["Suggested Job Role"])  y.head()  **#------------------Training and testing with Decision Tree----------------#**  **#------importing modules---------------#**  from sklearn import tree  from sklearn.model\_selection import train\_test\_split  from sklearn import preprocessing  from sklearn.metrics import accuracy\_score  **#---------------specifying percentage of test data from whole data--------#**  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X1,y,test\_size=0.2,random\_state=10)  **#-----------------classifying with decision tree-----------------------#**  clf = tree.DecisionTreeClassifier()  clf = clf.fit(X\_train, y\_train)  from sklearn.metrics import confusion\_matrix,accuracy\_score  y\_pred = clf.predict(X\_test)  Y\_pred  **#----------------calculating confusion vector values matrix and accuracy-------------#**  cm = confusion\_matrix(y\_test,y\_pred)  accuracy = accuracy\_score(y\_test,y\_pred)  print("confusion matrics=",cm)  print(" ")  print("accuracy=",accuracy\*100)  **#--------performing decision tree classification with entropy as measure------------#**  clf\_entropy = tree.DecisionTreeClassifier(criterion = "entropy", random\_state = 10)  clf\_entropy.fit(X\_train, y\_train)  entropy\_y\_pred=clf\_entropy.predict(X\_test)  cm\_entopy = confusion\_matrix(y\_test,entropy\_y\_pred)  entropy\_accuracy = accuracy\_score(y\_test,entropy\_y\_pred)  print("confusion matrix=",cm\_entopy)  print(" ")  print("accuracy=",entropy\_accuracy\*100)  **#------classification with svm------------------------#**  from sklearn import svm  clf = svm.SVC()  clf.fit(X\_train, y\_train) #------giving test data as input----#  svm\_y\_pred = clf.predict(X\_test) #--------doing prediction-----#  **#-----------calculating confusion matrix and accuracy----------#**  svm\_cm = confusion\_matrix(y\_test,svm\_y\_pred)  svm\_accuracy = accuracy\_score(y\_test,svm\_y\_pred)  print("confusion matrix=",svm\_cm)  print(" ")  print("accuracy=",svm\_accuracy\*100)  **#--------------classification using xgboost--------------#**  X\_train,X\_test,y\_train,y\_test=train\_test\_split(X1,y,test\_size=0.3,random\_state=10)  X\_train.shape  **#------------converting values of training and testing data into int64 datatype-------#**  X\_train=pd.to\_numeric(X\_train.values.flatten())  X\_train=X\_train.reshape((14000,38))  **#-------------importing and defining xgboost functions-----#**  from xgboost import XGBClassifier  model = XGBClassifier()  **#-----------training and testing with xg boost------#**  model.fit(X\_train, y\_train)  xgb\_y\_pred = clf.predict(X\_test)  **#-----calculating confusion matrix and accuracy after boosting--------#**  xgb\_cm = confusion\_matrix(y\_test,xgb\_y\_pred)  xgb\_accuracy = accuracy\_score(y\_test,xgb\_y\_pred)  print("confusion matrix=",xgb\_cm)  print(" ")  print("accuracy=",xgb\_accuracy\*100) |