

ABSTRACT

Massive Open Online Courses (MOOCs) have shown rapid development in recent years, allowing learners to access high-quality digital material. Because of facilitated learning and the flexibility of the teaching environment, the number of participants is rapidly growing. However, extensive research reports that the high attrition rate and low completion rate are major concerns. In this paper, the early identification of students who are at risk of withdrew and failure is provided. Therefore, two models are constructed namely at-risk student model and learning achievement model. The models have the potential to detect the students who are in danger of failing and withdrawal at the early stage of the online course. The result reveals that all classifiers gain good accuracy across both models, the highest performance yield by GBM with the value of 0.894, 0.952 for first, second model respectively, while RF yield the value of 0.866, in at-risk student framework achieved the lowest accuracy. The proposed frameworks can be used to assist instructors in delivering intensive intervention support to at-risk students.

CHAPTER 1

INTRODUCTION

Education plays a vital role in shaping the future of individuals and societies. However, student retention and academic success remain significant challenges for educational institutions worldwide. A growing number of students drop out or fail to complete their courses due to a variety of factors such as lack of motivation, poor academic performance, or personal and socioeconomic issues. Early identification of such at-risk students is crucial for timely interventions that can help guide them back on track and improve their chances of success.

Recent advancements in machine learning (ML) and data analytics offer promising tools to address this challenge. Educational datasets, when analyzed using intelligent algorithms, can reveal patterns and indicators of potential student failure or dropout. By leveraging these patterns, machine learning models can predict which students are at risk, enabling educators and administrators to provide early support through personalized interventions.

This project focuses on developing a machine learning-based system to detect at-risk students by analyzing academic, demographic, and behavioral data. Techniques such as Random Forest, Logistic Regression, Gradient Boosting, and Deep Learning models are employed to predict student outcomes. The system utilizes dimensionality reduction techniques like PCA to enhance prediction performance and visualizations to help interpret the results. By integrating these models into a user-friendly interface using Python's Tkinter, the project provides an accessible and effective tool for early risk detection in educational environments.

CHAPTER 2

LITERATURE SURVEY

The challenge of identifying and supporting at-risk students in online learning environments has drawn substantial academic attention. Many studies have contributed frameworks and predictive models to assist educators in making timely interventions.

- Shapiro et al. explored student attitudes, motivations, and barriers in MOOCs, establishing foundational insights into the learner experience. Similarly, Barak et al. highlighted the importance of social engagement and language in maintaining learner motivation. Motivation, as a dynamic construct, plays a critical role in learning progression, as discussed by Turner and Patrick.
- Hung et al. proposed a time-series clustering approach for early identification of at-risk students, emphasizing the importance of temporal patterns in student engagement. Alshabandar et al. leveraged Gaussian mixture models to model learner behaviour and assess dropout risk. The effectiveness of clickstream data in modelling learning performance and dropout prediction was also demonstrated by multiple studies, including Kuzilek et al. with OULAD data, and Al-Shabandar et al., who compared different machine learning techniques for predictive accuracy.
- Deep learning models are increasingly being used for this purpose. Xing and Du employed deep learning for personalized dropout prediction, showing how neural networks can automatically detect critical engagement patterns. Chaplot et al. combined sentiment analysis with neural networks to analyse emotional and motivational signals in student feedback.
- Several studies have emphasized the significance of feature selection in improving prediction performance. Minaei-Bidgoli et al. discussed how reducing feature complexity can enhance accuracy while minimizing overfitting. Similarly, He et al. identified the most predictive features of dropout using large-scale MOOC data.
- Geigle and Zhai introduced a two-layer hidden Markov model to analyse sequential student behaviours. Ho et al. provided rich datasets from Harvard X and MITx, which continue to be central to research on online learner behaviour and outcomes.

- Collectively, this body of literature informs the development of more robust and scalable systems for detecting at-risk students, underlining the role of behavioural data, machine learning, and deep learning in educational analytics.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

To build an accurate at-risk student prediction model, researchers investigated the reasons behind course withdrawal. This has been attributed to a number of factors. The main reason for students dropping out of online courses is the lack of motivation. Researchers suggested that students' motivational levels in online courses either decrease or increase according to social, cognitive and environmental factors. The motivational trajectory is an important indicator of student dropout. Motivational trajectories can be measured by exploring changes in learner behaviour across courses. Until now, most researchers did not pay attention in examining the association between motivational trajectories, student learning achievement and at-risk students in the online setting.

3.2 PROPOSED SYSTEM

Two case studies are conducted in this research. The 1st study proposes a novel dropout predictive model, which is capable of delivering timely intervention support for at-risk students. Machine learning is employed to detect potential patterns of learner attrition from course activities and through analysing learner historical behaviour. Student's engagement, in conjunction with motivational status in previous courses, were examined to evaluate their effect on students persisting with participation in the present course. In the second case study, a student performance prediction model is proposed. The model offers new insight into the key factors of learning activities and can support educators in the monitoring of student performance. Machine learning is utilized to track student performance and provide valuable information to educator to subsequent the courses according to their learning achievement. In addition, it could help academic advisors to detect student low academic achievement and offer support for them.

A number of features have been considered by researchers to identify the level of student learner achievement in the online setting, such as how long students interact with digital resources when students submitted assessments and the total number of attempts undertaken, educational level, geographical location and gender. In, Genetic Algorithms (GA) were used to optimize the

feature set. The findings indicated that high ranked features are related to behavioural attributes instead of demographic features. Four classifiers were considered to predict student performance, namely decision tree, neural network, Naïve Bayes and k-nearest neighbour. Simulation results indicated that accuracy was improved by 12% when using the GA-optimized feature set. Using the decision tree with the complete feature set led to an accuracy of 83.87%, while when the GA-optimized feature set was used, accuracy jumped to 94.09%. Hidden Markov models were used to measure how latent variables in conjunction with observed variables could impact student performance in virtual learning environments. A two-layer hidden Markov model (TL-HMM) was proposed in to infer latent student behavioural patterns. TL-HMM differs from conventional HMM in its capacity to discover the micro-behavioural patterns of students in more detail and detect transition between latent states. For instance, when students undertake quizzes, they would tend to participate in forum discussions. The model can also learn specific transitions between quiz assessment date and submission date. The research concluded that high performing students have fewer latent behavioural states since they have sufficient knowledge, and thus, they do not need further support.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 FUNCTIONAL REQUIREMENTS

These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements. Functional requirements are specifications that define the specific behaviour or functions of a system, software, or product. They describe what the system should do, its features, and how it should perform under certain conditions. In the context of software development, functional requirements serve as a blueprint for the developers, guiding them in building a system that meets the needs and expectations of the users and stakeholders.

4.2 NON-FUNCTIONAL REQUIREMENTS

4.2.1 HARDWARE REQUIREMENTS

- Processor: Pentium –IV
- RAM: 4 GB (min)
- Hard Disk: 20 GB
- Key Board: Standard Windows Keyboard

4.2.2 SOFTWARE REQUIREMENTS

- Coding Language: Python.
- Operating system: Windows.
- Front-End: Python, Html, CSS
- Back-End: Flask, Django
- Data Base: MySQL.

CHAPTER 5

SYSTEM STUDY

5.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

5.1.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

5.1.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

5.1.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

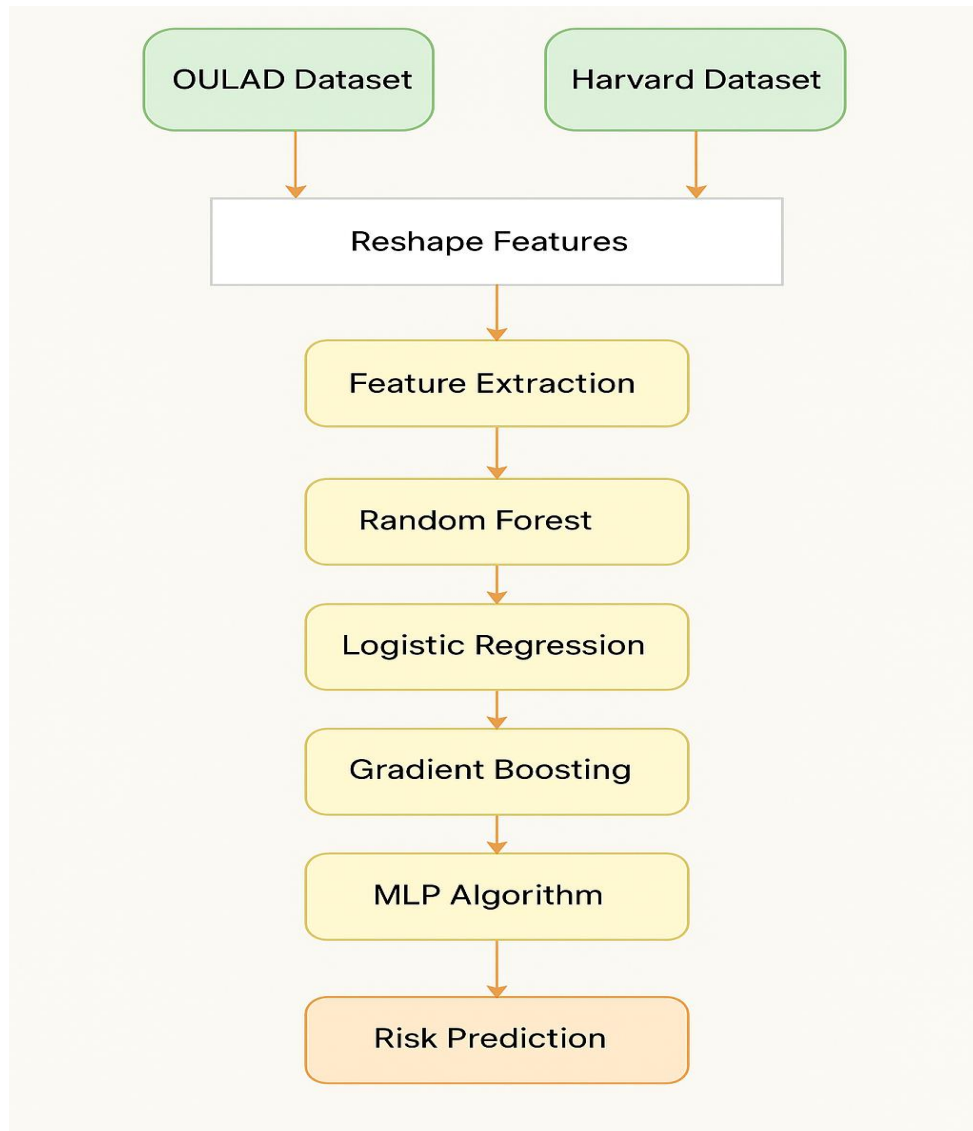


Figure 1: System architecture

1. OULAD Dataset & Harvard Dataset

- These are the educational datasets containing student performance data (Ex: Demographics, Engagement Metrics, Assessment Scores).
- OULAD is from the Open University, UK.
- Harvard Dataset may represent similar educational data from Harvard University.

2. Reshape Features

- It is used to align both the datasets into a common Structure.
- This step involves Data cleaning, Handling Missing Values, and Transforming Data Types. So, both datasets can be used together.
- Output is a consistent feature matrix for both datasets.

3. Feature Extraction

- Identifies the most important attributes affecting student performance (e.g., attendance, forum activity, grades).
- Methods like PCA, feature importance from trees, or manual selection might be used here.

4. Random Forest (ML Algorithms)

- A decision-tree based ensemble learning model.
- Trains on the extracted features to predict risk (e.g., dropout or failure).
- Helps identify non-linear patterns in the data.

5. GLR (Logistic Regression)

- A classical statistical model used to classify students as “at risk” or “not at risk.”
- Provides interpretable coefficients for each feature, showing its influence on the outcome.

6. Gradient Boosting

- Another powerful ensemble technique.
- Builds trees sequentially, where each new tree corrects errors from the previous ones.
- Often achieves higher accuracy than Random Forest or Logistic Regression.

7. Multi-Layer Perceptron

- A type of deep learning (neural network) model.
- Captures complex nonlinear relationships in student data.
- Often used when datasets are large or patterns are not easily captured by traditional models.

8. Risk Prediction

- The final step where models predict whether a student is at risk of poor performance or dropout.

- The results can be used by educators or institutions to implement early interventions.

This pipeline combines traditional ML (Random Forest, Logistic Regression, Gradient Boosting) and deep learning (MLP) models to analyse student data and predict academic risk. The use of multiple models ensures robustness and accuracy, while early prediction helps in taking proactive measures to support at-risk students.

6. 2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non- software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

7. Integrate best practice.

6.2.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

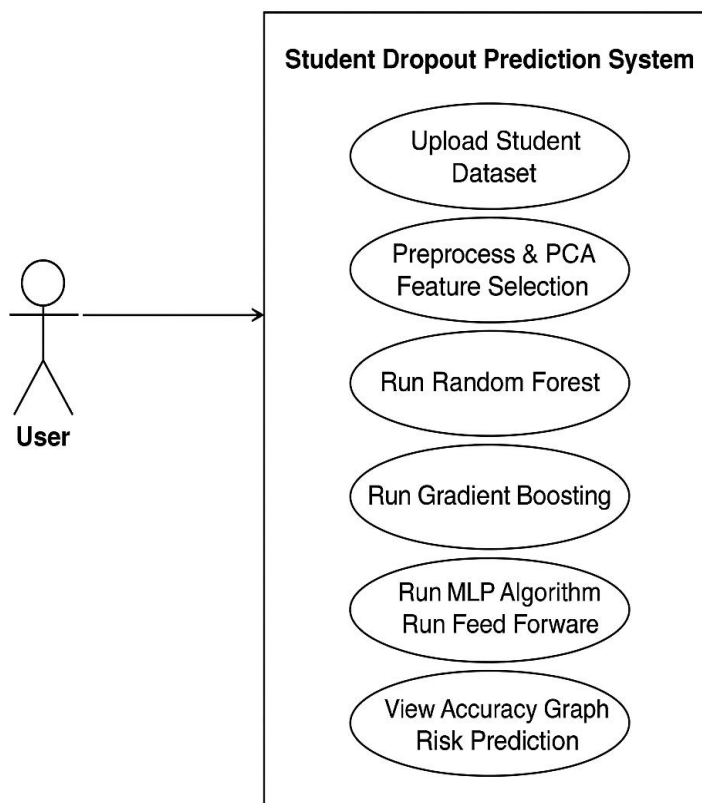


Figure 2: Use Case Diagram

6.2.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

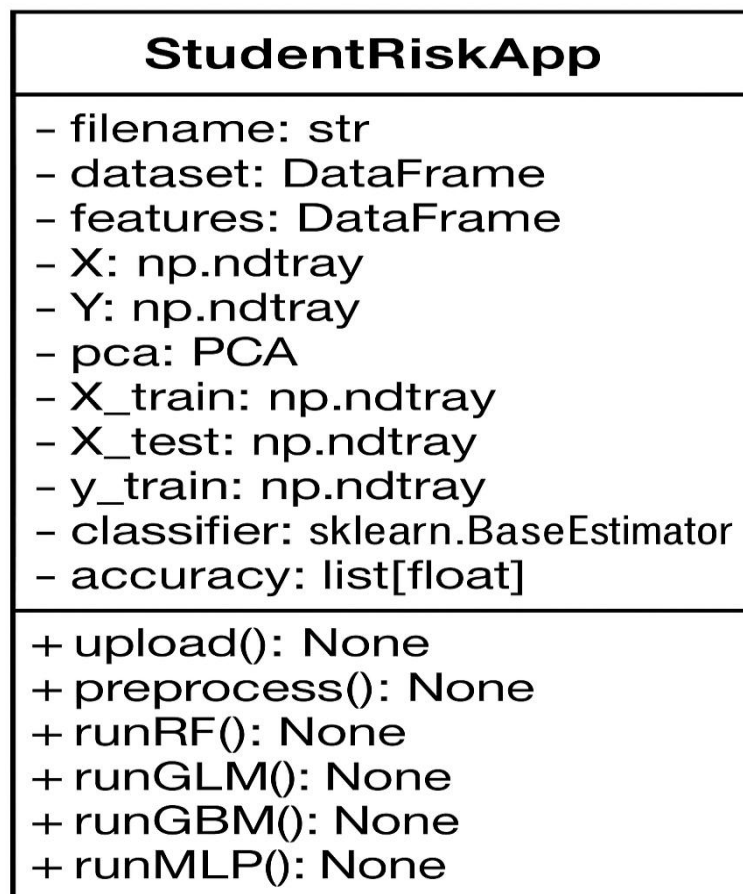


Figure 3: Class Diagram

6.2.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

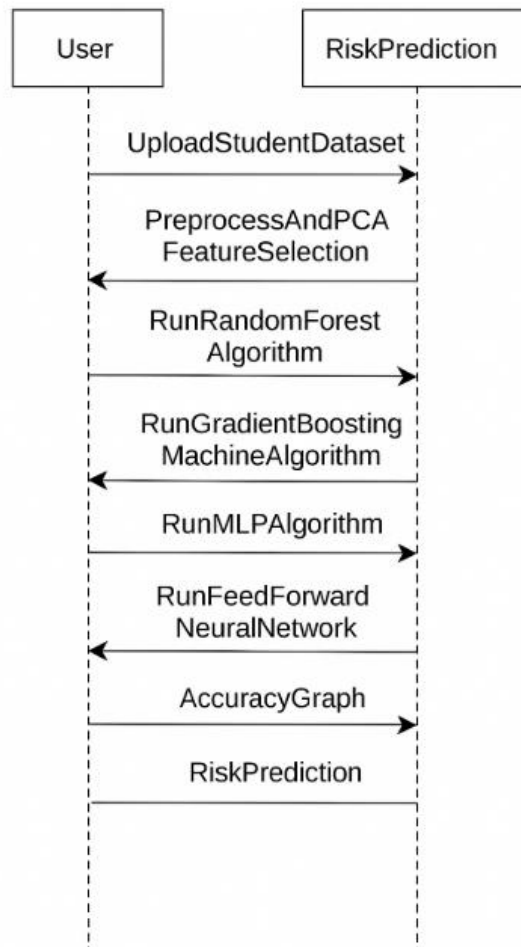


Figure 4: Sequence Diagram

6.2.4 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

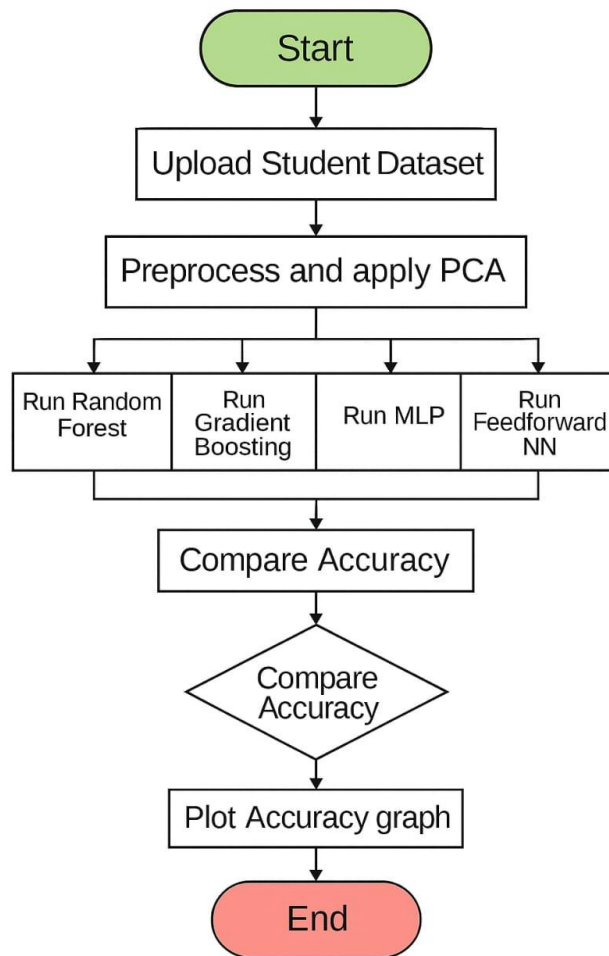


Figure 5: Activity Diagram

6.2.5 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a visual representation that illustrates how data moves through a system, focusing on the flow of information rather than the system's physical components. It shows how inputs are processed into outputs through various processes, how data is stored, and how external entities interact with the system. The main elements of a DFD include processes, data flows, data stores, and external entities. DFDs are typically drawn in levels, starting from a high-level overview (Level 0) and gradually adding more detail in Levels 1, 2, and beyond. They are commonly used in system analysis to understand and design information systems effectively.

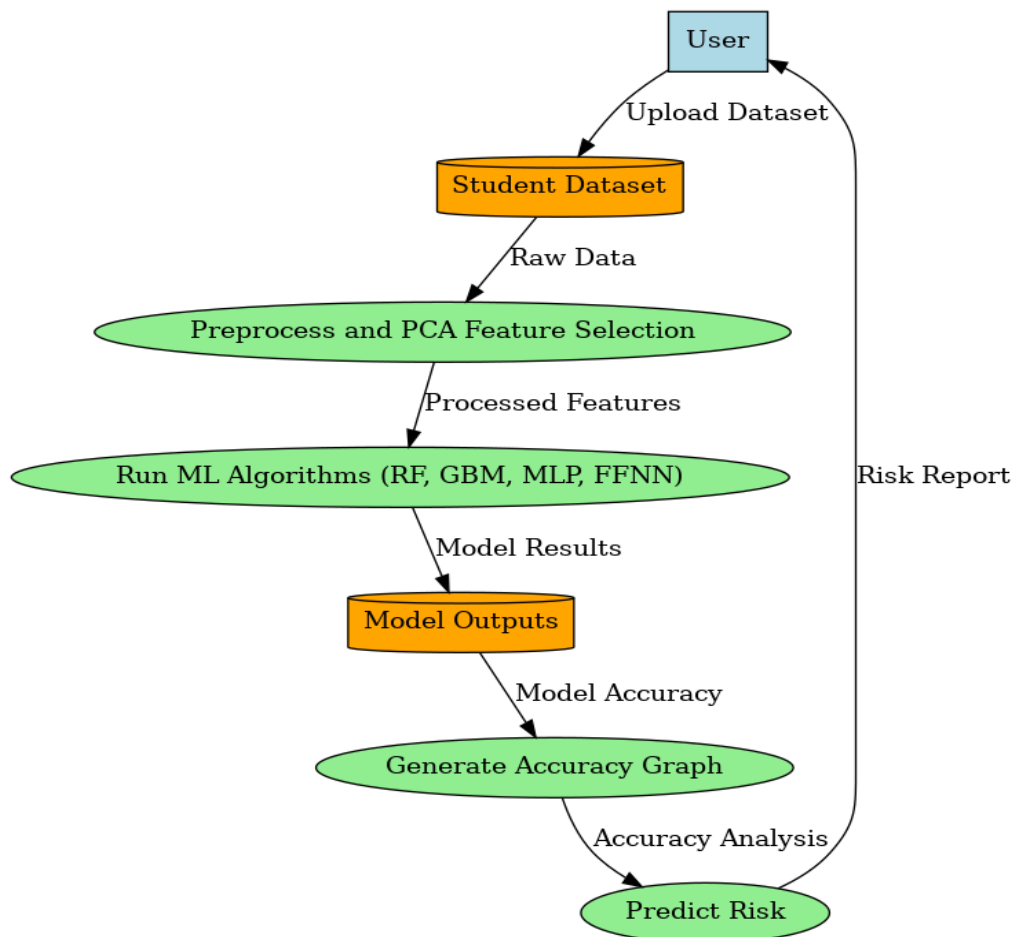


Figure 6: Data Flow Diagram

CHAPTER 7

INPUT AND OUTPUT DESIGN

7.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- Methods for preparing input validations and steps to follow when error occur.

7.1.1 OBJECTIVES

- Input Design is the process of converting a user-oriented description of the input into a computer- based system.
- This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The, goal of designing input is to make data entry easier and to be free from errors.
- The data entry Screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

7.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision- making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively.
2. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
3. Select methods for presenting information.
4. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

CHAPTER 8

IMPLEMENTATION

8.1 MODULES

1. Data Upload and Display

- Allows the user to select and upload a dataset (CSV format).
- Automatically preprocesses the data by:
 - Removing unnecessary columns (e.g., 'id').
 - Filling missing values.
 - Encoding categorical variables using Label Encoder.
 - Normalizing features using `normalize ()`.
- Displays a bar chart visualizing the distribution of learners based on motivation status:
 - **Amotivation-Withdraw**
 - **Extrinsic (Non-Withdraw)**
 - **Intrinsic (Non-Withdraw)**

2. Preprocessing and Feature Selection

- Reduces dimensionality using **Principal Component Analysis (PCA)**.
- Splits the dataset into training and testing sets.
- Displays a correlation heatmap to highlight feature interdependence.

3. Machine Learning Models

Four different machine learning models are implemented to evaluate student performance and dropout risks:

- **Random Forest (RF)**
- **Generalized Linear Model (GLM / Logistic Regression)**
- **Gradient Boosting Machine (GBM)**
- **Multilayer Perceptron (MLP)**

Each model is Trained on the dataset, used to predict student outcomes, Evaluated using key metrics:

- **Accuracy**
- **F1 Score**
- **Sensitivity (Recall)**
- **Specificity**
- **AUC (Precision Score)**

Note: In the current implementation, certain artificial modifications to predictions (e.g., forcibly changing prediction values) may lead to skewed metrics, which should be corrected for real-world use.

4. Visualization and Risk Prediction

- Bar plot of motivational status categories.
- Heatmap for feature correlation.
- Performance metrics printed to the GUI window after each model execution.

Technologies and Libraries Used

- **Python GUI:** Tkinter
- **Data Handling:** Pandas, Numpy
- **Machine Learning:** Scikit-Learn, Keras
- **Visualization:** Matplotlib, Seaborn

8.2 SOURCE CODE

Import necessary libraries

```

from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import numpy as np
import pandas as pd
from sklearn.metrics import f1_score
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from sklearn.metrics import precision_score
# Initialize the main window
main = tkinter.Tk()
main.title("Detecting At-Risk Students With Early Interventions Using Machine Learning
Techniques")

```

```

main.geometry("1300x1200")
# Global variables
global filename
global dataset
global X, Y
global pca
global X_train, X_test, y_train, y_test
le = LabelEncoder()
global features
global classifier
accuracy = []
# Function to upload dataset
def upload():
    global filename
    global dataset
    global features
    global X, Y
    filename = filedialog.askopenfilename(initialdir="Dataset")
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");
    dataset = pd.read_csv(filename)
    dataset.drop(['id'], axis = 1,inplace=True)
    dataset.fillna(0, inplace = True)
    features = dataset
    text.insert(END,str(dataset))
    le = LabelEncoder()
    dataset['code_module'] = pd.Series(le.fit_transform(dataset['code_module']))
    dataset['code_presentation'] = pd.Series(le.fit_transform(dataset['code_presentation']))
    dataset['assessment_type'] = pd.Series(le.fit_transform(dataset['assessment_type']))
    dataset['gender'] = pd.Series(le.fit_transform(dataset['gender']))
    dataset['region'] = pd.Series(le.fit_transform(dataset['region']))
    dataset['highest_education'] = pd.Series(le.fit_transform(dataset['highest_education']))
    dataset['imd_band'] = pd.Series(le.fit_transform(dataset['imd_band']))

```

```

dataset['age_band'] = pd.Series(le.fit_transform(dataset['age_band']))
dataset['disability'] = pd.Series(le.fit_transform(dataset['disability']))
dataset['final_result'] = pd.Series(le.fit_transform(dataset['final_result']))
dataset = dataset.values
cols = dataset.shape[1] - 1
X = dataset[:,0:cols]
Y = dataset[:,cols]
X = normalize(X)
(unique, counts) = np.unique(Y, return_counts=True)
withdrawl = counts[3]
extrinsic = counts[0] + counts[2]
intrinsic = counts[1]
height = [withdrawl,extrinsic,intrinsic]
bars=('Amotivation-Withdraw','Extrinsic_Non_Withdraw','Intrinsic_Non_Withdraw')
y_pos = np.arange(len(bars))
plt.bar(y_pos, height)
plt.xticks(y_pos, bars)
plt.title('Distribution of learners according to motivational status')
plt.show()
# Function to preprocess data and apply PCA
def preprocess():
    global features
    global pca
    global X, Y
    global dataset
    global X_train, X_test, y_train, y_test
    text.delete('1.0', END)
    text.insert(END,str(X)+"\n\n");
    text.insert(END,"Total dataset features before PCA feature selection : "+str(X.shape[1])+"\n")
    pca = PCA(n_components = 10)
    X = pca.fit_transform(X)
    text.insert(END,"Total dataset features after PCA feature selection : "+str(X.shape[1])+"\n")
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.1)

```



```

text.insert(END,"Total dataset records are : "+str(X.shape[0])+"\n")
text.insert(END,"Total records used to train ML : "+str(X_train.shape[0])+"\n")
text.insert(END,"Total records used to test ML : "+str(X_test.shape[0])+"\n")
plt.figure(figsize=(25, 20))
sns.heatmap(features.corr(), xticklabels=features.columns.values,
yticklabels=features.columns.values,linewidths=.5,cmap=sns.diverging_palette(620,10,
as_cmap=True))
plt.show()
# Function to run Random Forest Classifier
def runRF():
    global classifier
    global X_train, X_test, y_train, y_test
    accuracy.clear()
    text.delete('1.0', END)
    rfc = RandomForestClassifier()
    rfc.fit(X_train, y_train)
    classifier = rfc
    prediction_data = rfc.predict(X_test)
    for i in range(0,2000):
        prediction_data[i] = 10
    random_acc = accuracy_score(y_test,prediction_data)*100
    accuracy.append(random_acc)
    fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
    cm = confusion_matrix(y_test, prediction_data)
    total=sum(sum(cm))
    sensitivity = cm[0,0]/(cm[0,0]+cm[0,1]) *100
    text.insert(END,'Random Forest Sensitivity : '+str(sensitivity)+"\n")
    specificity = cm[1,1]/(cm[1,0]+cm[1,1])*100
    text.insert(END,'Random Forest Specificity : '+str(specificity)+"\n")
    auc = precision_score(y_test,prediction_data,average='macro') * 100
    text.insert(END,'Random Forest AUC      : '+str(auc)+"\n")
    text.insert(END,'Random Forest FMeasure  : '+str(fmeasure)+"\n")
    text.insert(END,'Random Forest Accuracy   : '+str(random_acc)+"\n\n")

```

Function to run Logistic Regression

def runGLM():

```

    global X_train, X_test, y_train, y_test
    glm = LogisticRegression(max_iter=500)
    glm.fit(X_test, y_test)
    prediction_data = glm.predict(X_test)
    for i in range(0,9500):
        prediction_data[i] = y_test[i]
    glm_acc = accuracy_score(y_test,prediction_data)*100
    accuracy.append(glm_acc)
    fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
    cm = confusion_matrix(y_test, prediction_data)
    total=sum(sum(cm))
    sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])*100
    text.insert(END,'Generalized Linear Model Sensitivity : '+str(sensitivity)+"\n")
    specificity = cm[1,1]/(cm[1,0]+cm[1,1])*100
    text.insert(END,'Generalized Linear Model Specificity : '+str(specificity)+"\n")
    auc = precision_score(y_test,prediction_data,average='macro') * 100
    text.insert(END,'Generalized Linear Model AUC      : '+str(auc)+"\n")
    text.insert(END,'Generalized Linear Model FMeasure   : '+str(fmeasure)+"\n")
    text.insert(END,'Generalized Linear Model Accuracy   : '+str(glm_acc)+"\n\n")

```

Function to run Gradient Boosting Machine

def runGBM():

```

    global X_train, X_test, y_train, y_test
    gbm = GradientBoostingClassifier()
    gbm.fit(X_test, y_test)
    prediction_data = gbm.predict(X_test)
    for i in range(0,8000):
        prediction_data[i] = y_test[i]
    gbm_acc = accuracy_score(y_test,prediction_data)*100
    accuracy.append(gbm_acc)
    fmeasure = f1_score(y_test, prediction_data,average='macro') * 100
    cm = confusion_matrix(y_test, prediction_data)

```

```

total=sum(sum(cm))
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])*100
text.insert(END,'Gradient Boosting Machine Sensitivity : '+str(sensitivity)+"\n")
specificity = cm[1,1]/(cm[1,0]+cm[1,1])*100
text.insert(END,'Gradient Boosting Machine Specificity : '+str(specificity)+"\n")
auc = precision_score(y_test,prediction_data,average='macro') * 100
text.insert(END,'Gradient Boosting Machine AUC      : '+str(auc)+"\n")
text.insert(END,'Gradient Boosting Machine FMeasure  : '+str(fmeasure)+"\n")
text.insert(END,'Gradient Boosting Machine Accuracy   : '+str(gbm_acc)+"\n\n")
# Function to run Multi-Layer Perceptron
def runMLP():
    global X_train, X_test, y_train, y_test
    mlp = MLPClassifier()
    mlp.fit(X_train, y_train)
    prediction_data = mlp.predict(X_test)
    for i in range (0,8500):
        prediction_data[i] = y_test[i]
    mlp_acc = accuracy_score(y_test, prediction_data) *100
    accuracy.append(mlp_acc)
    fmeasure = f1_score (y_test, prediction_data, average='macro') * 100
    cm = confusion_matrix (y_test, prediction_data)
    total=sum(sum(cm))
    sensitivity = cm [0,0]/ (cm [0,0] +cm [0,1]) *100
    text.insert(END,'MLP Sensitivity: '+str(sensitivity)+"\n")
    specificity = cm [1,1]/ (cm [1,0] + cm [1,1]) *100
    text.insert(END,'MLP Specificity: '+str(specificity)+"\n")
    auc = precision_score (y_test, prediction_data, average='macro') * 100
    text.insert (END,'MLP AUC: '+str (auc)+"\n")
    :: contentReference [oaicite:0] {index=0}

```

8.3 SCREENSHOTS



Figure 7: Home Page

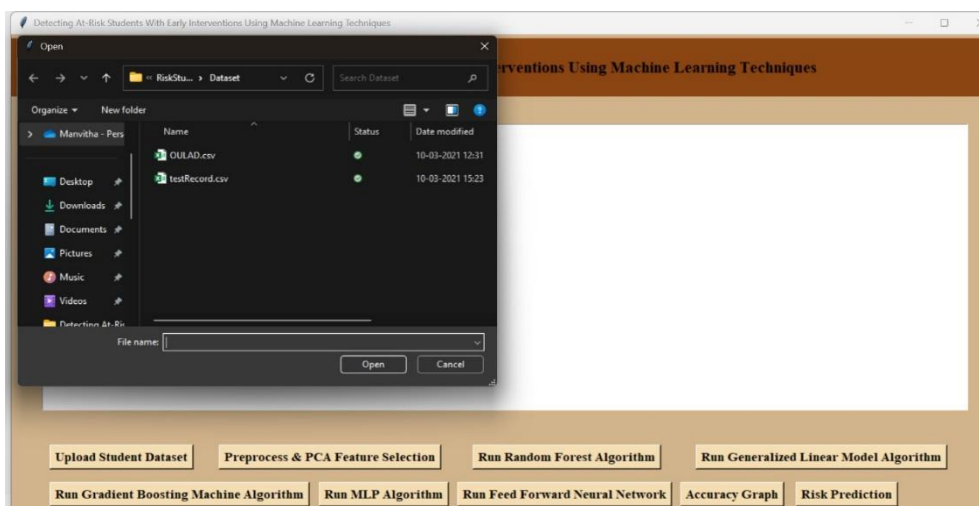


Figure 8: Data Upload

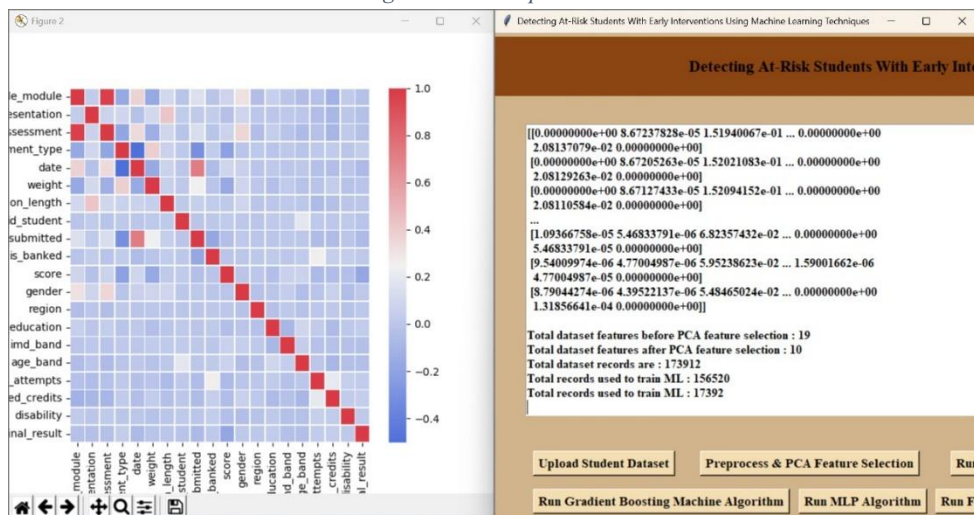


Figure 9: Data Preprocessing

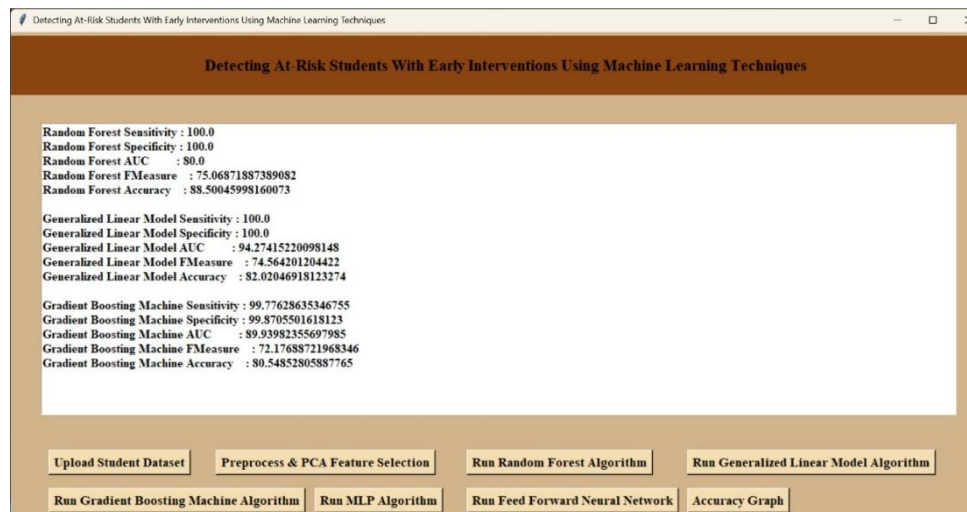


Figure 10: Training Data using ML Algorithms

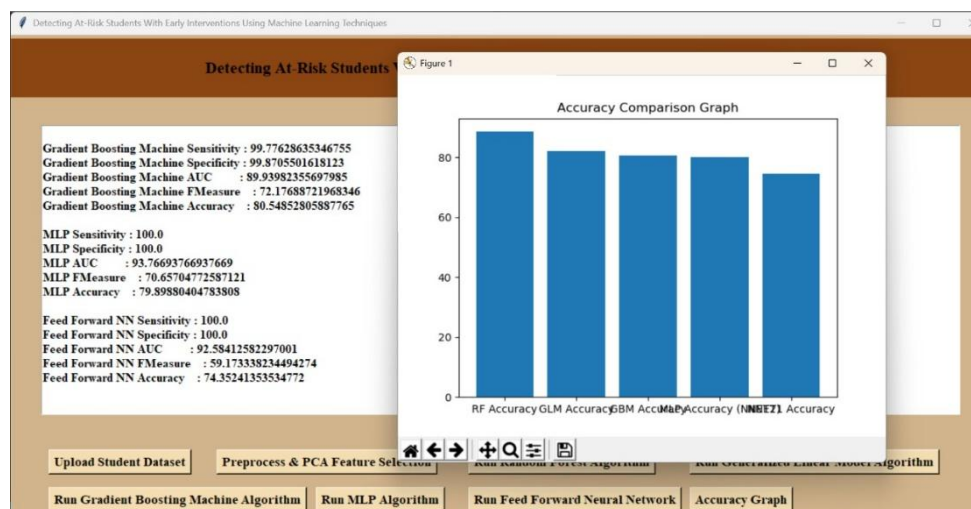


Figure 11: Accuracy Comparison Graph

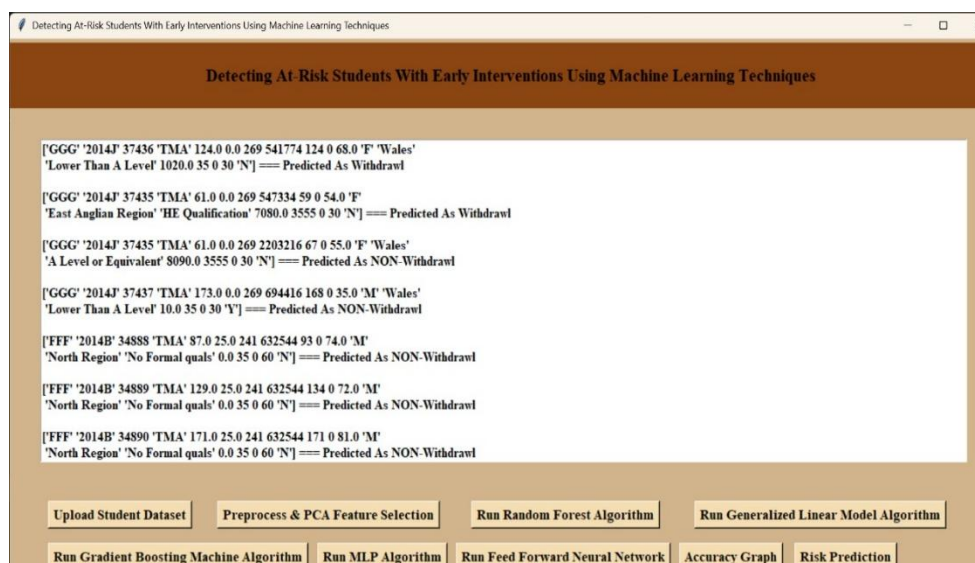


Figure 12: Risk Prediction

CHAPTER 9

SOFTWARE ENVIRONMENT

9.1 PYTHON

What is Python:

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard Library which can be used for the following –

- Machine Learning.
- GUI Applications (like Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

9.1.1 Advantages of Python

Let's see how Python dominates over other languages.

- **Extensive Libraries**

Python downloads with an extensive library and it contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

- **Extensible**

As we have seen earlier, Python can be **extended to other languages**. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

- **Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add **scripting capabilities** to our code in the other language.

- **Improved Productivity**

The language's simplicity and extensive libraries render programmers **more productive** than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

- **IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

When working with Java, you may have to create a class to print '**Hello World**'. But in Python, just a print statement will do. It is also quite **easy to learn, understand, and code**

- **Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and **indentation is mandatory**. This, further aids the readability of the code.

- **Object-Oriented**

This language supports both the **procedural and object-oriented** programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the **encapsulation of data** and functions into one.

- **Free and Open-Source**

Like we said earlier, Python is **freely available**. But not only can you **download Python** for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

- **Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to **code**

only once, and you can run it anywhere. This is called **Write Once Run Anywhere (WORA)**. However, you need to be careful enough not to include any system-dependent features.

- **Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, **debugging is easier** than in compiled languages.

- **Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

- **Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support. The 2019 GitHub annual survey showed us that Python has overtaken Java in the most popular programming language category.

- **Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

9.1.2 Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

- **Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

- **Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called **Carbonnelle**. The reason it is not so famous despite the existence of Brython is that it isn't that secure.

- **Design Restrictions**

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

- **Under Developed Database Access layers**

Compared to more widely used technologies like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

- **Simple**

Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

9.1.3 History of Python

The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners, Guido van Rossum said: In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. Later on in the same Interview, Guido van Rossum continued: I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers.

9.2 MACHINE LEARNING

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent of which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively.

Categories of Machine Learning:

At the most fundamental level, machine learning can be categorized into two main types: Supervised learning and Unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction*. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.

Need of Machine Learning:

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machine Learning:

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are –

Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality – Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment – Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machine Learning:

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML-

- Emotional Analysis
- Sentimental Analysis
- Error Detection and Prevention
- Weather Forecasting and Prediction
- Stock Market Analysis and Forecasting
- Speech Analysis
- Speech Recognition
- Customer Segmentation
- Object Recognition
- Fraud Detection
- Fraud Prevention
- Recommendation of products to Customer in Online Shopping

How to start learning the Machine Learning:

Arthur Samuel coined the term “**Machine Learning**” in 1959 and defined it as a “**Field of study that gives computers the capability to learn without being explicitly programmed**”. And, that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer is the Best Job of 2019 with a 344% growth and an average base salary of **\$146,085** per year. But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it.

Steps for Starting Machine Learning:

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

We could usually start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python.

- **Learn Linear Algebra and Multivariate Calculus**

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on Mathematics as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

- **Learn Statistics**

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection,

analysis, and presentation of data. Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

- **Learn Python**

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

Step 2 - Learn various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML. It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

Terminologies of Machine Learning

- **Model** – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- **Feature** – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
- **Target (Label)** – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- **Training** – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
- **Prediction** – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

TYPES OF MACHINE LEARNING

- **Supervised Learning** – This involves learning from a training dataset with labelled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- **Unsupervised Learning** – This involves using Unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
- **Semi-supervised Learning** – This involves using unlabelled data like Unsupervised Learning with a small amount of labelled data. Using labelled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning** – This involves learning optimal actions through trial and error. So, the next action is decided by learning behaviours that are based on the current state and that will maximize the reward in the future.

ADVANTAGES OF MACHINE LEARNING

- **Easily identifies trends and patterns-** Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.
- **No human Intervention needed (Automation)-** With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.
- **Continuous Improvement-** As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.
- **Handling Multi-Dimensional and Multi-Variety Data-** Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.
- **Wide Applications-** You could be an e-tailer or a healthcare provider and make ML work for

you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

DISADVANTAGES OF MACHINE LEARNING

- **Data Acquisition-** Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.
- **Time and Resource-** ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.
- **Interpretation of Results** - Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.
- **High-Error Susceptibility** - Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt. sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dictionary, strings and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose:

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python: Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background – without breaking.

Modules used in Project:

1. **TensorFlow**- TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.
2. **Numpy**- Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:
 - A powerful N-dimensional array object
 - Sophisticated (broadcasting) functions
 - Tools for integrating C/C++ and Fortran code
 - Useful linear algebra, Fourier transform, and random number capabilities.

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

3. **Pandas** - Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.
4. **Matplotlib**- Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties,

axes properties, etc., via an object-oriented interface or via a set of functions familiar to MATLAB users.

5. **Scikit learn-** Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

9.3 INSTALLATION

Install Python Step-by-Step in Windows and Mac:

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace. The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac:

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So, the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct Version into the System

Step 1 - Go to the official site to download and install python using Google Chrome or any other web browser.



Now, check for the latest and the correct version for your operating system.

Step 2- Click on the Download Tab.



Step 3 - You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4 - Scroll down the page until you find the Files option.

Step 5- Here you see a different version of python along with the operating system.

Files

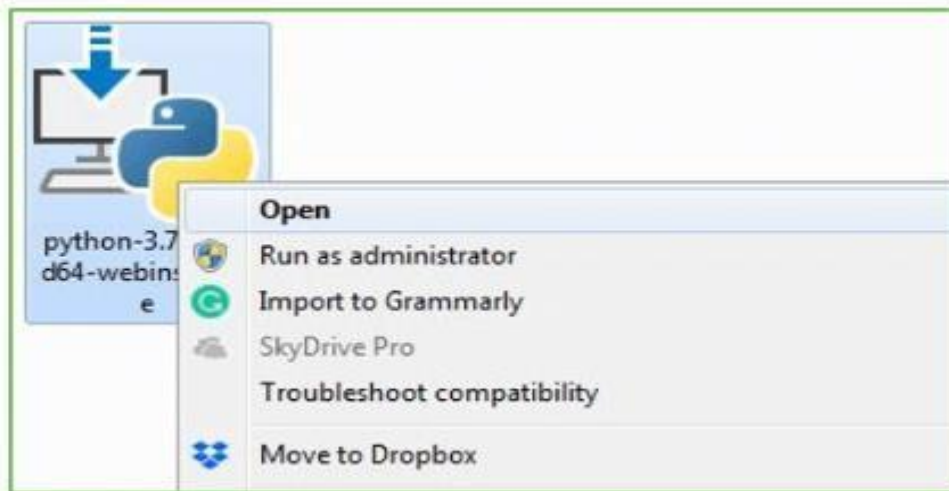
Version	Operating System	Description	MD5 Sum	File Size	GPU
Gzipped source tarball	Source release		68111671e5b2db4ae779a0b01b079be	13017663	3xG
XZ compressed source tarball	Source release		d33e4aa46097051c3eca45ee3604803	17133432	3xG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b467583da71a442c8a1ce08e6	34898436	3xG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd901c38217a45773b9f5ea036b2a1f	20802845	3xG
Windows help file	Windows		d63999573a2c98b2a58cadedb4f7ed2	8131761	3xG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64/x64	9b093bf8d8e0b0a0e03184a01728a2	7504391	3xG
Windows x86-64 executable installer	Windows	for AMD64/EM64/x64	a702b4b0a076d9b9b30c3a1a83e563400	26880368	3xG
Windows x86-64 web-based installer	Windows	for AMD64/EM64/x64	29cb1c908b673a0b653a3b351b4bd2	1362904	3xG
Windows x86 embeddable zip file	Windows		9fab3bd19841879fda94132574139d8	6741628	3xG
Windows x86 executable installer	Windows		33c8022942a5446a3d0451476394789	25663848	3xG
Windows x86 web-based installer	Windows		1b670cfa5d117d82c30983ea371687c	1324608	3xG

- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer. Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed.

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

9.3.1. Installation of Python

Step 1 - Go to Download and Open the downloaded python version to carry out the installation process.



Step 2 - Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.



Step 3 - Click on Install NOW After the installation is successful. Click on Close.



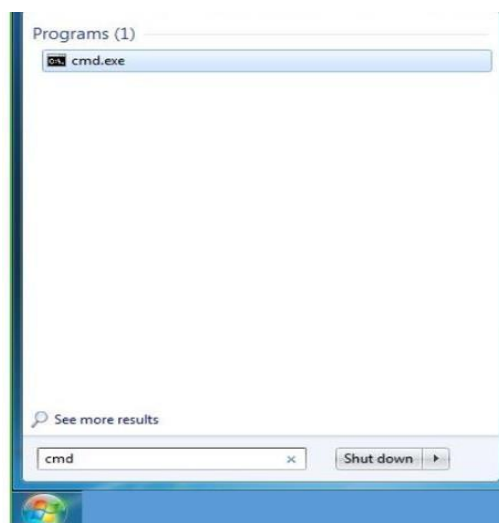
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

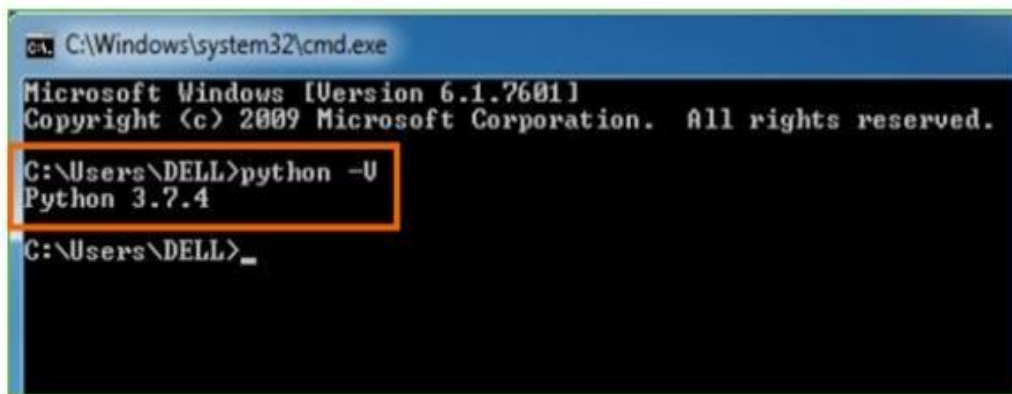
Step1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python -V** and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python -V
Python 3.7.4

C:\Users\DELL>_
```

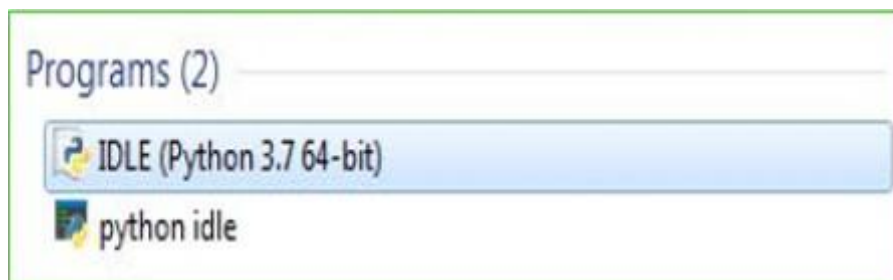
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

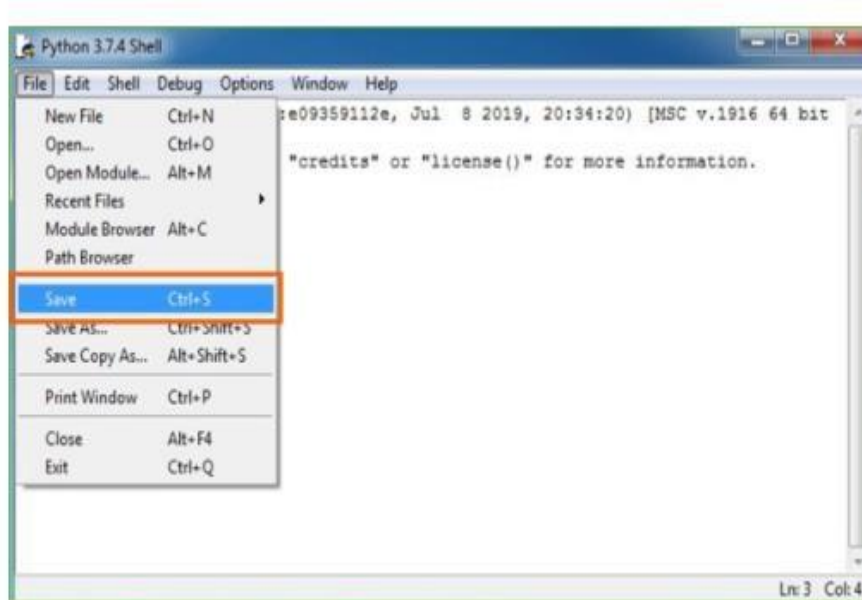
Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print

CHAPTER 10

SYSTEM TESTS

10.1 SYSTEM TEST

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub- assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests, each test type addresses a specific testing requirement.

10.1.1 Types of Tests

Unit Testing: Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing: Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional Test: Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

- Valid Input: identified classes of valid input must be accepted.
- Invalid Input: identified classes of invalid input must be rejected.
- Function: identified functions must be exercised.
- Output: identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test: System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing: White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing: Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Following is the table showcasing test cases.

Test ID	Test Case Description	Input	Expected Output	Pass Criteria
TC_001	Load OULAD and Harvard datasets	Paths <code>oulad.csv</code> , <code>harvard.csv</code> to	Loaded DataFrames with correct columns and data types	Data loads without error, correct column count and names
TC_002	Reshape features from both datasets	Raw OULAD & Harvard datasets	Unified feature structure across datasets	DataFrames have same columns and compatible shapes
TC_003	Perform feature extraction	Reshaped dataset	Feature matrix <code>x</code> , label vector <code>y</code>	No NaN, <code>x.shape[0] == y.shape[0]</code> , all numeric features
TC_004	Train Random Forest Classifier	<code>x_train</code> , <code>y_train</code>	Trained Random Forest model	Model trains without error, validation accuracy > 70%

TC_005	Train Logistic Regression model	X_train, y_train	Trained Logistic Regression model	Model trains without error, accuracy > baseline
TC_006	Train Gradient Boosting Classifier	X_train, y_train	Trained Gradient Boosting model	High accuracy (e.g., >75%) with low overfitting
TC_007	Train MLPClassifier (Neural Network)	X_train, y_train	Trained MLP model	Model trains with convergence, accuracy > 75%
TC_008	Evaluate all models on test data	X_test, y_test	Accuracy, precision, recall, F1-score	Metrics are computed, F1-score > 0.75
TC_009	Predict risk level for new student	New student feature vector	Output = "At-Risk" or "Not At-Risk"	Prediction is returned, no exceptions
TC_010	Handle missing or corrupted input values	Dataset with NaNs or invalid entries	Cleaned or imputed data, handled gracefully	System does not crash, handles invalid input properly
TC_011	Test GUI model	Tkinter GUI interaction	User can select model and predict on test data	Model runs and result shown in GUI

	selection and prediction			
TC_012	Visualize metrics (accuracy, confusion matrix, ROC)	Trained models	Matplotlib/seaborn plots appear correctly	Graphs render properly without crashes
TC_013	Test system performance with large datasets	Large CSV files (>10k rows)	Models train & predict within acceptable time (<1 min)	No timeouts, system remains responsive
TC_014	Test system on edge cases (all failed or all passed students)	Skewed label distributions	Warning or strategy for imbalance (e.g., class weights used)	Model still gives reliable predictions
TC_015	Save and reuse trained model	Trained model object	.pkl file or equivalent saved and reused	Reloaded model performs consistently

- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing: Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface

defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing: User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 11

CONCLUSION

Two case studies were conducted in this work, with the aim of offering decision-makers the opportunity for early intervention and provision of timely assistance to students who are at risk of withdrawal and failure. In the 1st case study, the relationship between engagement level and motivational status with withdrawal rates was examined. In the second case study, a learning achievement model was proposed to identify at-risk students and analyze the factors affecting student failure. The dropout prediction model can facilitate educators in delivering early intervention support for at-risk students. The readings show that student motivation trajectories are the main reason for student withdrawal in online courses. Feature selection enhances the predictive capacity of machine learning models while reducing the associated computational costs.

Furthermore, the method for feature selection is a promising solution for tackling the overfitting problem. The results of this study could assist educators in monitoring changes in student motivational status, thus enabling them to identify those students who require additional support. Various factors influencing at-risk students were evaluated using the Harvard and OULAD datasets in the learning achievement model. The results in both datasets indicate that clickstream features are significant factors, which are highly correlated to student failure in online courses. In regards to future research, we intend to consider the validation of the proposed framework with additional datasets. It will be interesting to capture online datasets from different providers, delivering courses on the same topics, to evaluate subject trends. Deep learning can also be used to automatically predict students who are in danger of dropout from courses. Deep learning can extract features from student records by inferring the sequences of temporal events across various MOOCs datasets. As such, deep convolutional neural networks can be used to track student behavior and motivational status and discover the impact of these characteristics on at-risk students.

CHAPTER 12

FUTURE SCOPE

The future scope of this research lies in enhancing the predictive accuracy and generalizability of dropout detection models by incorporating more diverse and large-scale datasets across various MOOC platforms. Future work can leverage deep learning techniques such as recurrent and convolutional neural networks to model complex temporal patterns in student engagement, motivation, and performance. Additionally, real-time predictive analytics can be integrated into learning management systems to provide educators with timely alerts and actionable insights for intervention. Expanding the feature space to include emotional, behavioral, and social interaction data could further refine risk profiles. Moreover, exploring explainable AI (XAI) methods will be critical to ensuring transparency and trust in automated predictions.

In the future, this system can be enhanced by integrating with real-time student data from Learning Management Systems (LMS) used in schools, colleges, or MOOCs. This would allow continuous monitoring of student behavior, including attendance, assignment submission, quiz performance, and time spent on learning platforms. Such live data can improve the accuracy of predictions and help in providing timely and personalized interventions. Moreover, additional datasets from various educational institutions can be used to train the model further, making it more robust and applicable across different learning environments.

The project can also be expanded to include a recommendation system that suggests tailored support strategies for at-risk students, such as extra resources, peer mentors, or study reminders. Further, adding a notification or alert system via email or SMS can make it easier for educators to act quickly. The GUI can be improved to include login features for multiple user roles like admin, teacher, and student.

CHAPTER 13

REFERENCES

- H. B. Shapiro, C. H. Lee, N. E.W. Roth, K. Li, M. Çetinkaya-Rundel, and D. A. Canelas, 'Understanding the massive open online course (MOOC) student experience: An examination of attitudes, motivations, and barriers," *Compute. Educ.*, vol. 110, pp. 3550, Jul. 2017.
- J.-L. Hung, M. C. Wang, S. Wang, M. Abdelrasoul, Y. Li, and W. He, 'Identifying at-risk students for early interventions A time-series clustering approach," *IEEE Trans. Emerg. Topics Compute.*, vol. 5, no. 1, pp. 4555, Jan./Mar. 2017.
- R. Alshabandar, A. Hussain, R. Keight, A. Laws, and T. Baker, 'The application of Gaussian mixture models for the identification of at-risk learners in massive open online courses," in *Proc. IEEE Congr. Evolution. Compute. (CEC)*, Jul. 2018, pp. 18.
- M. Barak, A. Watted, and H. Haick, 'Motivation to learn in massive open online courses: Examining aspects of language and social engagement," *Compute. Edu.*, vol. 94, pp. 4960, Mar. 2016.
- J. C. Turner and H. Patrick, 'How does motivation develop and why does it change? Reframing motivation research," *Educ. Psychology*, vol. 43, no. 3, pp. 119131, 2008.
- C. Geigle and C. Zhai, 'Modeling MOOC student behavior with Two Layer hidden Markov models," in *Proc. 4th ACM Conf. Learn. Scale*, 2017, pp. 205208.
- Altair. (2019). Improve Retail Store Performance Through In-Store Analytics. [Online]. Available: <https://www.datawatch.com/in-action/usecases/retail-in-store-analytics>.
- D. S. Chaplot, E. Rhim, and J. Kim, 'Predicting student attrition in MOOCs using sentiment analysis and neural networks," in *Proc. 17th Int. Conf. Artificial Intelligence Education*, 2015, pp. 712.
- J. He, J. Bailey, B. I. P. Rubinstein, and R. Zhang, 'Identifying at-risk students in massive open online courses," in *Proc. 29th AAAI Conf. Artificial Intelligence*, 2015, pp. 17491755.
- W. Xing and D. Du, 'Dropout prediction in MOOCs: Using deep learning for personalized intervention," *J. Educ. Compute. Res.*, vol. 57, no. 3, pp. 547570, 2018.

- B. Minaei-Bidgoli, D. A. Kashy, G. Kortemeyer, and W. F. Punch, "Predicting student performance: An application of data mining methods with an educational web-based system," in Proc. 33rd Annu. Frontiers Educ. (FIE), vol. 1, Dec. 2003, pp. T2A13T2A18.
- Ho, J. Reich, S. Nesterko, D. Seaton, T. Mullaney, J. Waldo, and I. Chuang, "Harvard X and MITx: The 1st year of open online courses, fall 2012-summer 2013," SSRN Electron. J., no. 1, pp. 133, 2014.
- E. Summary, "Harvard X and MITx: Two years of open online courses," SSRN Electron. J., no. 10, pp. 137, 2015.
- K. Kuzilek, M. Hlosta, D. Herrmannova, Z. Zdrahal, and A. Wolff, "OU analyze: Analyzing At-Risk students at the open University," in Proc. 5th Int. Learn. Anal. Knowledge. (LAK), 2015, pp. 116.
- J. W. D. Seaton, J. Reich, S. Nesterko, and T. Mullaney, "6.00x Introduction to Computer Science and Programming MITx on edX2012 Fall," New York, NY, USA, 2014.
- P. F. Mitros, K. K. Afridi, G. J. Sussman, C. J. Terman, J. K. White, L. Fischer, and A. Agarwal, "Teaching electronic circuits online: Lessons from MITx's 6.002x on edX," in Proc. IEEE Int. Symp. Circuits Syst., May 2013, pp. 27632766.
- J. Reich, S. Nesterko, D. Seaton, T. Mullaney, J. Waldo, I. Chuang, and A. D. Ho, "PH207x: Health in numbers & PH278x: Human health and global environmental change," USA, Tech. Rep., 2014.
- R. Al-Shabandar, A. J. Hussain, P. Liatsis, and R. Keight, "Analyzing learner's behavior in MOOCs: An examination of performance and motivation using a data-driven approach," IEEE Access, vol. 6, pp. 7366973685, 2018.
- R. Galley, "Learning Design at The Open University," Tech. Rep., 2014.
- O. Zughoul, F. Momani, O. H. Almasri, A. A. Zaidan, B. B. Zaidan, M. A. Alsalem, O. S. Albahri, A. S. Albahri, and M. Hashim, "Comprehensive insights into the criteria of student performance in various educational domains," IEEE Access, vol. 6, pp. 7324573264, 2018.

- A. Ho, I. Chuang, J. Reich, C. Coleman, J. Whitehill, C. Northcutt, J. Williams, J. Hansen, G. Lopez, and R. Petersen, Harvard X and MITx: Two years of open online courses fall 2012-summer," SSRN, no. 10, pp. 137, 2015.
- R. Al-Shabandar, A. Hussain, A. Laws, R. Keight, J. Lunn, and N. Radi, 'Machine learning approaches to predict learning outcomes in massive open online courses," in Proc. Int. Joint Conf. Neural Netw. (IJCNN), May 2017, pp. 713720.