**Result Analysis of Student data**

**Abstract**

A desktop GUI application is created to analyse the Results Data provided. This data contains the details of students USN, Subject Code, Marks obtained in CIE, SEE and the total, along with few inferences whether the student has passed or failed in a particular subject and Grade obtained, Date and Type of the exam. We analyse this data to infer a few more details. Few of our objectives include:

Visualisation of the student performance in a particular subject based on the total marks obtained.

Prediction of the performance of a particular student based on the subject code (output- Grade) using machine learning.

To check whether a Student with a backlog clears the exam in successive attempts.

Visualisation of subcode versus number of students who pass or fail that subject.

Visualization of student performance based on a particular year.

Predicting the score of a student in the consecutive semester based on the score of the similar subject in present or previous semesters.

These objectives contain graphical representations and predictions using the given data.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **DESCRIPTION** | **PAGE** |
| **1** | **Introduction** |  |
| **2** | **Literature Survey** |  |
| **3** | **Requirements** |  |
| **4** | **About Spark** |  |
| **5** | **About PySpark** |  |
| **6** | **About Distributed File System** |  |
| **7** | **Dataset** |  |
| **8** | **Wire Frame Diagrams** |  |
| **9** | **Design** |  |
| **10** | **Development Details** |  |
| **11** | **Screenshots** |  |
| **12** | **Conclusion** |  |
| **13** | **References** |  |

**Introduction**

A Desktop application is created using Tkinter for GUI. Our application has a Login page to authorise the users. We provide six options or objectives for the users to choose from. These objectives are Visualisation of the performance of students in a particular subject based on total marks obtained. Prediction of the performance of a particular student based on the subject code (output- Grade) using machine learning. To check whether a Student with a backlog clears the exam in successive attempts. Visualisation of subcode versus number of students who pass or fail that subject. Visualization of student performance based on a particular year. Predicting the score of a student in the consecutive semester based on the score of the similar subject in present or previous semesters. On clicking any of these, another tab pops open for that particular objective. On providing the required information in the entry fields, the user obtains the desired output in the form of a graphical represention, Lists or values.

These objectives are implemented using the PySpark library as the given dataset is large. For objectives with graphical repesentations, we use Matplotlib library. For formating the dataset according to our requirement we use PySpark and Pandas.

**Literature survey**

**2.1 A Big Data Analysis Framework Using Apache Spark and Deep Learning**

**Author:** Anand Gupta, Hardeo Kumar Thakur, Ritvik Shrivastava, Pulkit Kumar, Sreyashi Nag

**Description:**

With the spreading prevalence of Big Data, many advances have recently been made in this field. Frameworks such as Apache Hadoop and Apache Spark have gained a lot of traction over the past decades and have become massively popular, especially in industries. It is becoming increasingly evident that effective big data analysis is key to solving artificial intelligence problems. Thus, a multi-algorithm library was implemented in the Spark framework, called MLlib. While this library supports multiple machine learning algorithms, there is still scope to use the Spark setup efficiently for highly timeintensive and computationally expensive procedures like deep learning. In this paper, we propose a novel framework that combines the distributive computational abilities of Apache Spark and the advanced machine learning architecture of a deep multilayer perceptron (MLP), using the popular concept of Cascade Learning. We conduct empirical analysis of our framework on two real world datasets. The results are encouraging and corroborate our proposed framework, in turn proving that it is an improvement over traditional big data analysis methods that use either Spark or Deep learning as individual elements.

**2.2 A Review Study of Apache Spark in Big Data Processing**

**Author:** V Srinivas Jonnalagadda, P Srikanth, Krishnamachari Thumati, Sri Hari Nallamala

**Description:**

Apache Spark has features: Speed, Ease of Use, Combines SQL, streaming, and complex analytics, Combines SQL, streaming, and complex analytics, Advanced Analytics, Runs Everywhere.

Spark in Big Data processing:

1. Spark Built on Hadoop: Standalone, Hadoop Yarn, Spark in MapReduce (SIMR)

2. Components of Spark: Apache Spark Core, Spark SQL, Spark Streaming, MLlib (Machine Learning Library), GraphX.

3. Apache Spark Operations: Resilient Distributed Datasets, Data Sharing is Slow in MapReduce, Iterative Operations on MapReduce, Interactive Operations on MapReduce, Data Sharing using Spark RDD, Iterative Operations on Spark RDD, Interactive Operations on Spark RDD.

**2.3 Predictive Modelling In IoT Using Apache Spark**

**Author:** Shikha Sonii, Amritpal Singh

**Description:**

Predictive Analytics models:

1. Binary classification model: Binary classification model is defined as splitting two classes in such a way that one class is considered a normal state whereas the other class is considered an abnormal state

2. Outlier detection model: Outlier detection model detects the values that are more likely to be separated from normal values and fall in a sparsely populated region (which indicate that any event has a low probability occurring in that area) and considered abnormal values.

3. Time series model: Time series model observes data over a period of time, making it time based. It is used to observe trends and patterns in a certain period of time to read the values and do changes accordingly to increase efficiency by organizations.

**2.4 A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools**

**Author:** D. P. Acharjya, Kauser Ahmed P

**Description:**

Challenges In Big Data Analytic: Data Storage and Analysis, Knowledge Discovery and Computational Complexities, Scalability and Visualization of Data, Information Security.

Open Research Issues In Big Data Analytics: IoT for Big Data Analytics, Cloud Computing for Big Data Analytics, Bio-inspired Computing for Big Data Analytics, Quantum Computing for Big Data Analysis.

Tools For Big Data Processing: Apache Hadoop and MapReduce, Apache Mahout, Apache Spark, Dryad, Storm, Apache Drill, Jaspersoft, Saspersoft.

**2.5 Machine Learning with PySpark - Review**

**Author:** Raswitha Bandi, J Amudhavel, R Karthik

**Description:**

A reasonable distributed memory-based Computing system for machine learning is Apache Spark. Spark is being superior in computing when compared with Hadoop. Apache Spark is a quick, simple to use for handling big data that has worked in modules of Machine Learning, streaming SQL, and graph processing. We can apply machine learning algorithms to big data easily, which makes it simple by using Spark and its machine learning library MLlib, even this can be made simpler by using the Python API PySpark. This paper presents the study on how to develop machine learning algorithms in PySpark.

**2.6 Big Data Analysis: Apache Spark Perspective**

**Author:** Abdul Ghaffar Shoro, Tariq Rahim Soomro

**Description:** Big Data have gained enormous attention in recent years. Analyzing big data is very common requirement today and such requirements become nightmare when analyzing of bulk data source such as twitter twits are done, it is really a big challenge to analyze the bulk amount of twits to get relevance and different patterns of information on timely manner. This paper will explore the concept of Big Data Analysis and recognize some meaningful information from some sample big data source, such as Twitter twits, using one of industries emerging tool, known as Spark by Apache.

**2.7 Integrating E-Governance with Big Data Analytics using Apache Spark**

**Author:** Poonam Salwan, Veerpaul Kaur Maan

**Description:** The constant innovations and rapid developments in the IT industry have revolutionized the thinking and mindset of the people throughout the world. Government departments have also been computerized to provide transparent, efficient and responsible government through e-governance. The government have been providing access to various websites or portal for filing complaints, uploading or downloading forms, pictures, data or PDFs to avail the government services. Enlightened citizens are frequently using the portal to access government services. Thus, the size and volume of data that need to be managed by government departments have been increasing drastically under e-governance. The traditional database management system is not designed to deal with such mix type of data. Moreover, the speed at which the e-governance generated data need to be processed is another big challenge being faced by traditional database system. All the abovesaid concerns can be solved by using the emerging technology - Big Data Analytics techniques. Big data analytic techniques can make the government more efficient and transparent by processing structured, unstructured or mixed types data at a great speed. In this paper, we shall understand the scenario for the need or the emergence of big data analytics in egovernance and knowhow of Apache Spark. This paper proposes a practical approach to integrate big data analytics with egovernance using Apache Spark. This paper also reflects how major issues of traditional database management system (mixed type datasets, speed and accuracy) can be resolved through the integration of big data analytics and e-governance.

**2.8 Big data analytics for smart sports using apache spark**

**Author:** Jacob Reece, SeongYong Hong

**Description:** On most collegiate sports teams’ injuries are plaguing rosters. The issue with the injuries on a football team is the amount of them that come from training overloads and lack of recovery from their training sessions. Coaches are not being able to take their team to their full potential and athletes are not being able to perform at their full potential as well. We are getting direct real-time data from wearable technology called Whoop. It collects the strain that the athlete goes through, their Heart Rate Variability (HRV), Respiration efficiency, and so many more things that could help improve the health of the athletes. Therefore, this paper will present the research proposal that the use of Big Data Analysis will be able to incorporate predictive analysis methods. With this proposal system in place coaches, athletes, and athletic trainers will be able to help prevent injuries and improve the overall performance of the team. By retrieving the real-time data from the Whoop band, we will be able to make real-time decisions and create correlations between the collected data and the performance of the athletes. The purpose of this proposed research is to create a system that will be beneficial to all types of sports teams, athletes, and university athletics budgets with big data analysis.

**2.9 Efficient Big Data Analysis with Apache Spark in HDFS**

**Author:** Amol Bansod

**Description:** With the size of data increasing each day, the traditional methods of data processing have become inefficient and time consuming. Today, Facebook, Google, Twitter are generating Petabytes of data each day. This large amount of data is given the term ‘Big Data’. To overcome this inefficiency, the processing of Data can be performed using Apache spark. Apache Spark is a fast, in-memory processing of large amount of data. In this research paper, the author discusses an efficient way of analyzing Big Data stored in Hadoop Distributed File System HDFS using Apache Spark framework, and its advantages over Hadoop MapReduce framework.

**2.10 Aadhaar Data Analysis Comparison in MapReduce, Hive and Spark**

**Author:** Roopa R, Varsha Ryali, Tejasvi Shrivastava, Syed Mahmood Nabeel Anwar

**Description:** Aadhaar with a 12-digit unique identification number of every Indian provides demographic and biometric information and is mandatory for various purposes like benefit transfer directly, healthcare, etc. Approximately Aadhaar details need to store 1.3 Billion Indians which attributes to the concept of big data. In this paper, the proposed hybrid model analyses the Aadhaar dataset w.r.t different research interrogations such as count of applicants based on gender, state-wise approved and by age type applicants. In the existing systems, Aadhaar data analyses are done either manually or in primitive SQL platforms which may take days to complete. In this paper, the focus is on Aadhaar data analysis using different distributed computing frameworks like MapReduce, Hive, and Apache Spark on top of Hadoop that could be used for the purpose of better decision-making by all government firms and we provide the valid conclusion that Apache Spark framework is efficient in terms of performance.

**Requirements**

**3.1 Hardware Requirements**

8 GB DDR4 RAM

Intel i5 Gen 7 processor

**3.2 Software Requirements**

Python

cycler 0.11.0

fonttools 4.32.0

kiwisolver 1.4.2

matplotlib 3.5.1

numpy 1.22.3

packaging 21.3

pandas 1.4.2

Pillow 9.1.0

py4j 0.10.9.3

pyparsing 3.0.8

pyspark 3.2.1

python-dateutil 2.8.2

pytz 2022.1

six 1.16.0

**Functional requirements**

The function requirements of this project include:

1. The application allows the user to register and authenticates before providing access.

2. The Result data is analysed and objectives have been identified.

3. The application facilitates the user to choose among the given objectives.

**Objective1:** Visualisation of the student performance in a particular subject based on the total marks obtained.

**Objective2:** Prediction of the performance of a particular student based on the subject code (output- Grade) using machine learning.

**Objective3:** To check whether a student with a backlog clears the exam in successive attempts.

**Objective4:** Visualisation of subcode versus number of students who pass or fail that subject.

**Objective5:** Visualization of student performance based on a particular year.

**Objective6:** Predicting the score of a student in the consecutive semester based on the score of the similar subject in present or previous semesters.

**Non Functional requirements**

The Non-function requirements of this project include:

1. **Reliability:** The system shall be completely operational.

2. **Usability:** The application has a user-friendly interface and can be comprehended by any user.

3. **Performance**: The application runs seamlessly in both Windows and Linus operating system. This application does not need internet connection to run.

4. **Security:** Only authenticated users are allowed to operate the application.

5. **Online User Documentation and Help:** The application can be easily available in GitHub.

**About Spark**

**About Spark**

Apache Spark is a fast and general-purpose cluster computing system. It provides high-level APIs in Scala, Java, and Python that make parallel jobs easy to write, and an optimized engine that supports general computation graphs. It also supports a rich set of higher-level tools including Shark (Hive on Spark), MLlib for machine learning, GraphX for graph processing, and Spark Streaming.

Apache Spark™ is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.

**Spark Core:**

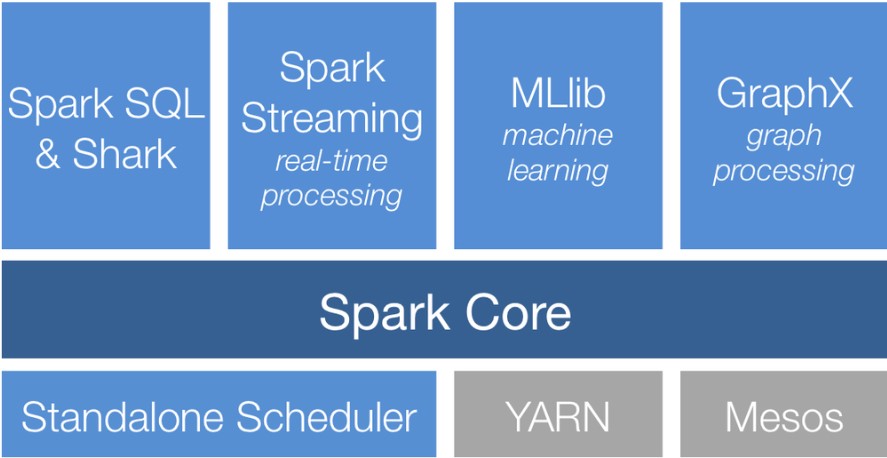
Spark Core contains the basic functionality of Spark, including components for task scheduling, memory management, fault recovery, interacting with storage systems, and more. Spark Core is also home to the API that defines Resilient Distributed Datasets (RDDs), which are Spark’s main programming abstraction. RDDs represent a collection of items distributed across many compute nodes that can be manipulated in parallel. Spark Core provides many APIs for building and manipulating these collections.

**Spark SQL:** Spark SQL provides support for interacting with Spark via SQL as well as the Apache Hive variant of SQL, called the Hive Query Language (HiveQL). Spark SQL represents database tables as Spark RDDs and translates SQL queries into Spark operations. Beyond providing the SQL interface to Spark, Spark SQL allows developers to intermix SQL queries with the programmatic data manipulations supported by RDDs in Python, Java and Scala, all within a single application. This tight integration with the rich and sophisticated computing environment provided by the rest of the Spark stack makes Spark SQL unlike any other open source data warehouse tool. Spark SQL was added to Spark in version 1.0. Shark is a project out of UC Berkeley that pre-dates Spark SQL and is being ported to work on top of Spark SQL. Shark provides additional functionality so that Spark can act as drop-in replacement for Apache Hive. This includes a HiveQL shell, as well as a JDBC server that makes it easy to connect external graphing and data exploration tools.

**Spark Streaming:** Spark Streaming is a Spark component that enables processing live streams of data. Examples of data streams include log files generated by production web servers, or queues of messages containing status updates posted by users of a web service. Spark Streaming provides an API for manipulating data streams that closely matches the Spark Core’s RDD API, making it easy for programmers to learn the project and move between applications that manipulate data stored in memory, on disk, or arriving in real-time. Underneath its API, Spark Streaming was designed to provide the same degree of fault tolerance, throughput, and scalability that the Spark Core provides.

**MLlib:** Spark comes with a library containing common machine learning (ML) functionality called MLlib. MLlib provides multiple types of machine learning algorithms, including binary classification, regression, clustering and collaborative filtering, as well as supporting functionality such as model evaluation and data import. It also provides some lower level ML primitives including a generic gradient descent optimization algorithm. All of these methods are designed to scale out across a cluster.

**GraphX:** GraphX is a library added in Spark 0.9 that provides an API for manipulating graphs (e.g., a social network’s friend graph) and performing graph-parallel computations. Like Spark Streaming and Spark SQL, GraphX extends the Spark RDD API, allowing us to create a directed graph with arbitrary properties attached to each vertex and edge. GraphX also provides set of operators for manipulating graphs (e.g., subgraph and mapVertices) and a library of common graph algorithms (e.g., PageRank and triangle counting).



**Key features of Spark:**

**Fast processing:** The most important feature of Apache Spark that has made the big data world choose this technology over others is its speed. Big data is characterized by its volume, variety, velocity, value, and veracity due to which it needs to be processed at a higher speed. Spark contains Resilient Distributed Datasets (RDDs) that save the time taken in reading and writing operations, and hence it runs almost 10–100 times faster than Hadoop.

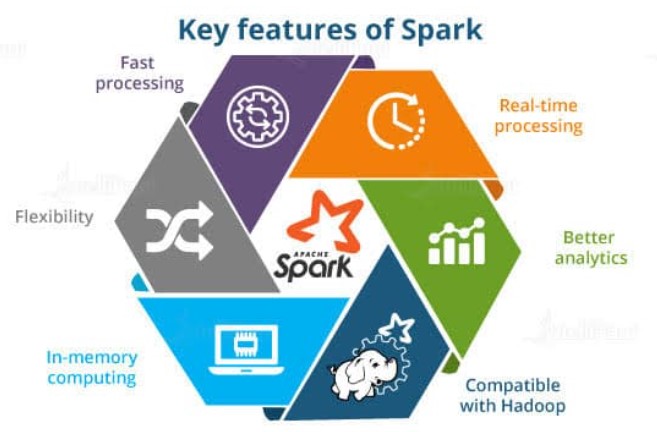
**Flexibility:** Apache Spark supports multiple languages and allows developers to write applications in Java, Scala, R, or Python. Equipped with over 80 high-level operators, this tool is quite rich from this aspect.

**In-memory computing:** Spark stores data in the RAM of servers, which allows it to access data quickly, and in-turn this accelerates the speed of analytics.

**Real-time processing:** Spark is able to process real-time streaming data. Unlike MapReduce, which processes the stored data, Spark is able to process the real-time data and hence is able to produce instant outcomes.

**Better analytics:** Contrasting to MapReduce that includes Map and Reduce functions, Spark has much more in store. Apache Spark comprises a rich set of SQL queries, Machine Learning algorithms, complex analytics, etc. With all these Spark functionalities, Big Data Analytics can be performed in a better fashion.

**Compatibility with Hadoop:** Spark is not only able to work independently; it can work on top of Hadoop as well. Not just this, it is certainly compatible with both versions of the Hadoop ecosystem.



**Ecosystem**

Apache Spark™ integrates with your favorite frameworks, helping to scale them to thousands of machines.

**Data science and Machine learning**



**SQL analytics and BI**



**Storage and Infrastructure**



**Benefits of Apache Spark**

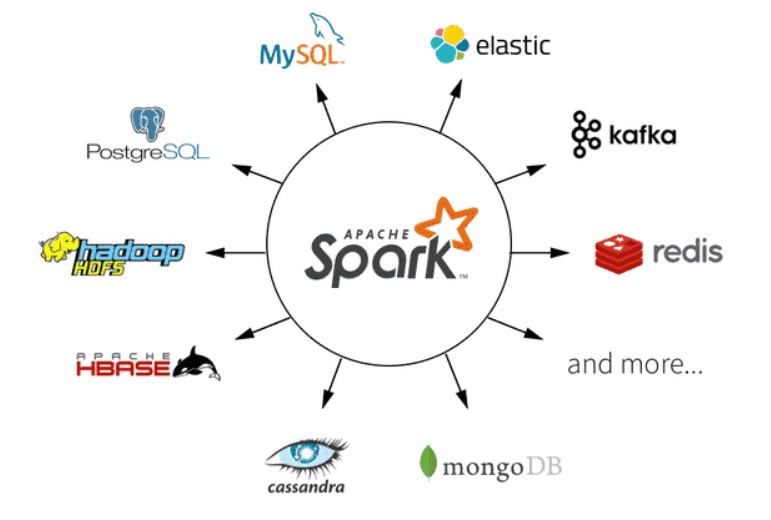
**Speed:** Engineered from the bottom-up for performance, Spark can be 100x faster than Hadoop for large scale data processing by exploiting in memory computing and other optimizations. Spark is also fast when data is stored on disk, and currently holds the world record for large-scale on-disk sorting

**Ease of Use:** Spark has easy-to-use APIs for operating on large datasets. This includes a collection of over 100 operators for transforming data and familiar data frame APIs for manipulating semi-structured data.

**A Unified Engine:** Spark comes packaged with higher-level libraries, including support for SQL queries, streaming data, machine learning and graph processing. These standard libraries increase developer productivity and can be seamlessly combined to create complex workflows.

Apache Spark: The largest open source project in data processing.

Since its release, Apache Spark, the unified analytics engine, has seen rapid adoption by enterprises across a wide range of industries. Internet powerhouses such as Netflix, Yahoo, and eBay have deployed Spark at massive scale, collectively processing multiple petabytes of data on clusters of over 8,000 nodes. It has quickly become the largest open source community in big data, with over 1000 contributors from 250+ organizations.

****

**About PySpark**

**About PySpark**

PySpark is an interface for Apache Spark in Python. It not only allows you to write Spark applications using Python APIs, but also provides the PySpark shell for interactively analyzing your data in a distributed environment.

PySpark is the Python API for Apache Spark, an open source, distributed computing framework and set of libraries for real-time, large-scale data processing. PySpark is a good language to learn to create more scalable analyses and pipelines.

Apache Spark is basically a computational engine that works with huge sets of data by processing them in parallel and batch systems. Spark is written in Scala, and PySpark was released to support the collaboration of Spark and Python. In addition to providing an API for Spark, PySpark helps you interface with Resilient Distributed Datasets (RDDs) by leveraging the Py4j library.

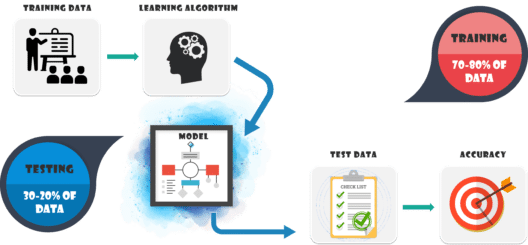
The key data type used in PySpark is the Spark dataframe. This object can be thought of as a table distributed across a cluster, and has functionality that is similar to dataframes in R and Pandas. If you want to do distributed computation using PySpark, then you’ll need to perform operations on Spark dataframes and not other Python data types.

**Py4J** is a popular library which is integrated within PySpark and allows Python to dynamically interface with JVM (Java Virtual Machine) objects. PySpark features quite a few libraries for writing efficient programs. Furthermore, there are various external libraries that are also compatible, including:

**PySparkSQL** - A PySpark library to apply SQL-like analysis on a huge amount of structured or semi-structured data. You can also use SQL queries with PySparkSQL.



**MLlib** - A wrapper over PySpark and Spark’s machine learning (ML) library. MLlib supports many machine learning algorithms for classification, regression, clustering, collaborative filtering, dimensionality reduction, and underlying optimization primitives.



**GraphFrames** - A graph processing library that provides a set of APIs for performing graph analysis efficiently, using the PySpark core and PySparkSQL. It is optimized for fast distributed computing.

**Key features of PySpark**

There are various features of the PySpark which are given below:

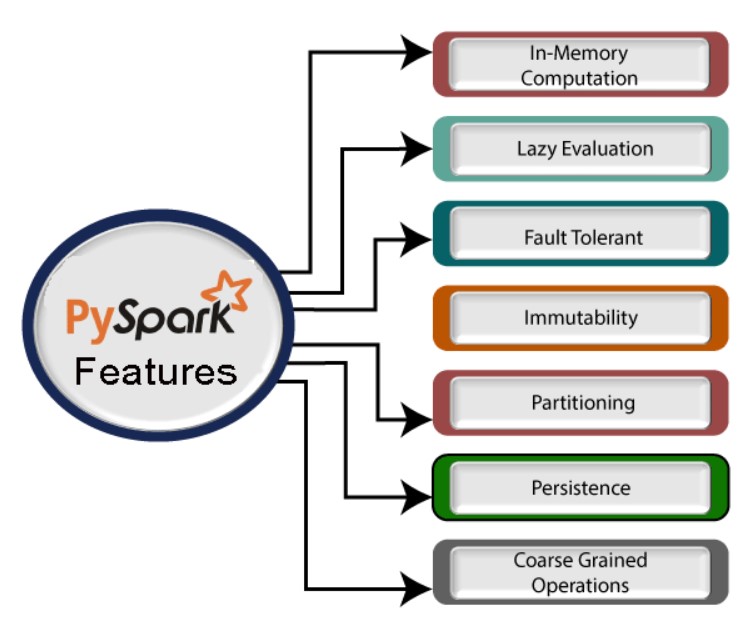
**Real-time Computation:** PySpark provides real-time computation on a large amount of data because it focuses on in-memory processing. It shows the low latency.

**Support Multiple Language:** PySpark framework is suited with various programming languages like Scala, Java, Python, and R. Its compatibility makes it the preferable frameworks for processing huge datasets.

**Caching and disk constancy**: PySpark framework provides powerful caching and good disk constancy.

**Swift Processing:** PySpark allows us to achieve a high data processing speed, which is about 100 times faster in memory and 10 times faster on the disk.

**Works well with RDD:** Python programming language is dynamically typed, which helps when working with RDD. We will learn more about RDD using Python in the further tutorial.



A large amount of data is generated offline and online. These data contain the hidden patterns, unknown correction, market trends, customer preference and other useful business information. It is necessary to extract valuable information from the raw data.

We require a more efficient tool to perform different types of operations on the big data. There are various tools to perform the multiple tasks on the huge dataset but these tools are not so appealing anymore. It is needed some scalable and flexible tools to crack big data and gain benefit from it.



**Real-life usage of PySpark**

Data is an essential thing for every industry. Most of the industries works on big data and hires analysts to extract useful information from the raw data. Let's have a look at the impact of the PySpark on several industries.

1. **Entertainment Industry**

The entertainment industry is one of the largest sectors which is growing towards online streaming. The popular online entertainment platform Netflix uses the Apache spark for real-time processing to personalized online movies or web series to its customers. It processes approx. 450 billion events per day that are streamed on server-side application.

2. **Commercial Sector**

The commercial sector also uses Apache Spark's Real-time processing system. Banks and other financial fields are using Spark to retrieve the customer's social media profile and analyze to gain useful insights which can help to make the right decision.

The extracted information is used for the credit risk assessment, targeted ads, and customer segmentation.

Spark plays a significant role in Fraud Detection and widely used in machine learning tasks.

3. **Healthcare**

Apache Spark is used to analyze the patient records along with the previous medical reports data to identify which patient is probable to face health issues after being discharged from the clinic.

4. **Trades and E-commerce**

The leading e-commerce websites like Flipkart, Amazon, etc, use Apache Spark for targeted advertising. The other websites such as Alibaba provides targeted offers, enhanced customer experience and optimizes overall performance.

5. **Tourism Industry**

The tourism industry widely uses Apache Spark to provide advice to millions of travelers by comparing hundreds of tourism websites.

**About distributed file system**

**About distributed file system**

A Distributed File System (DFS) as the name suggests, is a file system that is distributed on multiple file servers or multiple locations. It allows programs to access or store isolated files as they do with the local ones, allowing programmers to access files from any network or computer.

The main purpose of the Distributed File System (DFS) is to allows users of physically distributed systems to share their data and resources by using a Common File System. A collection of workstations and mainframes connected by a Local Area Network (LAN) is a configuration on Distributed File System. A DFS is executed as a part of the operating system. In DFS, a namespace is created and this process is transparent for the clients.

DFS has two components:

Location Transparency – Location Transparency achieves through the namespace component.

Redundancy – Redundancy is done through a file replication component.

**Dataset**

**Result Data**

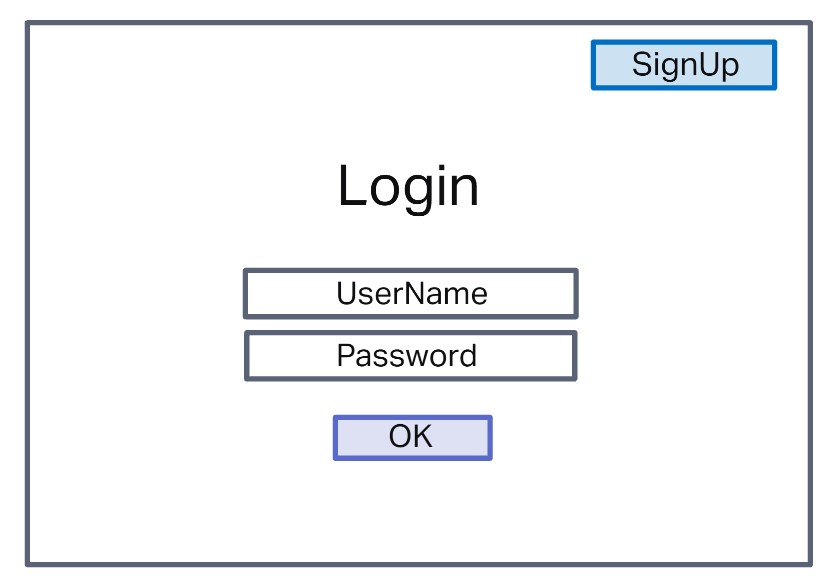
Sample Data- First 10 rows have been displayed

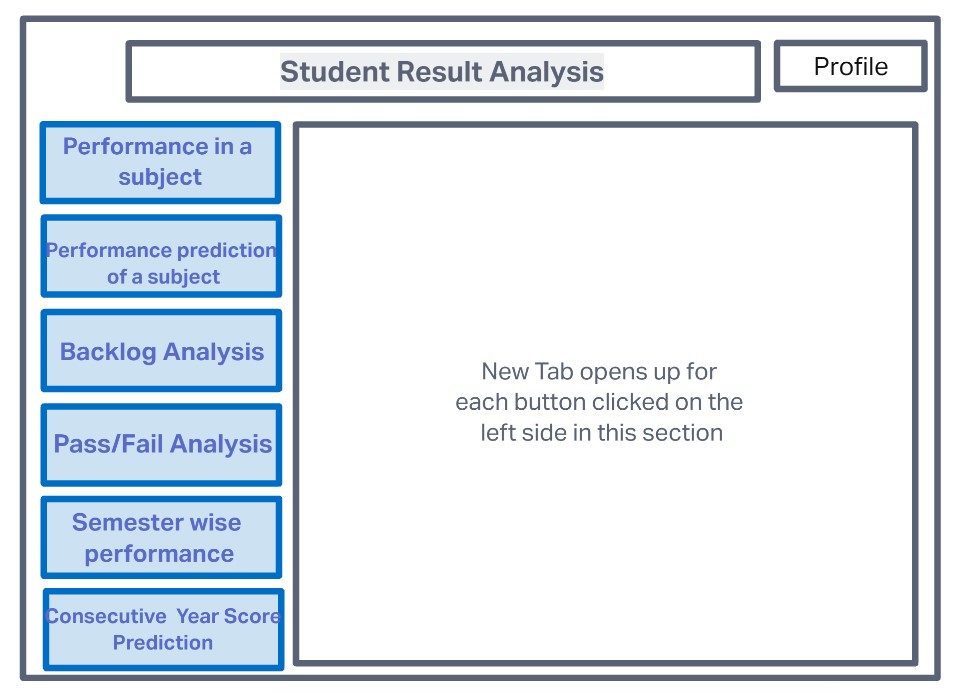
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| USN | SCODE | CIE | SEE | TOT | RESULT | GRADE | DATE | ExamType |
| 4JN18CS001 | 18MAT11 | 32 | 41 | 73 | P | 8 | 12019 | Regular |
| 4JN18CS001 | 18PHY12 | 29 | 47 | 76 | P | 8 | 12019 | Regular |
| 4JN18CS001 | 18ELE13 | 25 | 43 | 68 | P | 7 | 12019 | Regular |
| 4JN18CS001 | 18CIV14 | 31 | 26 | 57 | P | 6 | 12019 | Regular |
| 4JN18CS001 | 18EGDL15 | 31 | 56 | 87 | P | 9 | 12019 | Regular |
| 4JN18CS001 | 18PHYL16 | 32 | 40 | 72 | P | 8 | 12019 | Regular |
| 4JN18CS001 | 18ELEL17 | 34 | 52 | 86 | P | 9 | 12019 | Regular |
| 4JN18CS001 | 18EGH18 | 27 | 32 | 59 | P | 6 | 12019 | Regular |
| 4JN18CS002 | 18MAT11 | 33 | 51 | 84 | P | 9 | 12019 | Regular |
| 4JN18CS002 | 18PHY12 | 36 | 45 | 81 | P | 9 | 12019 | Regular |

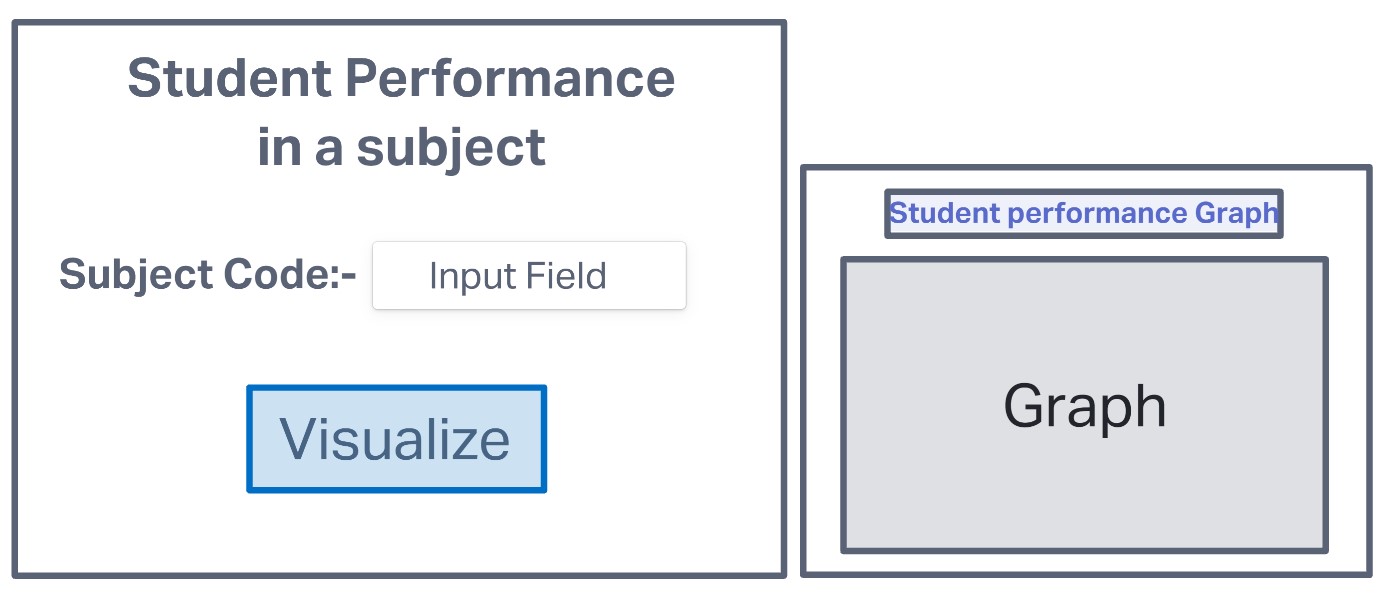
Table 1: Results\_data

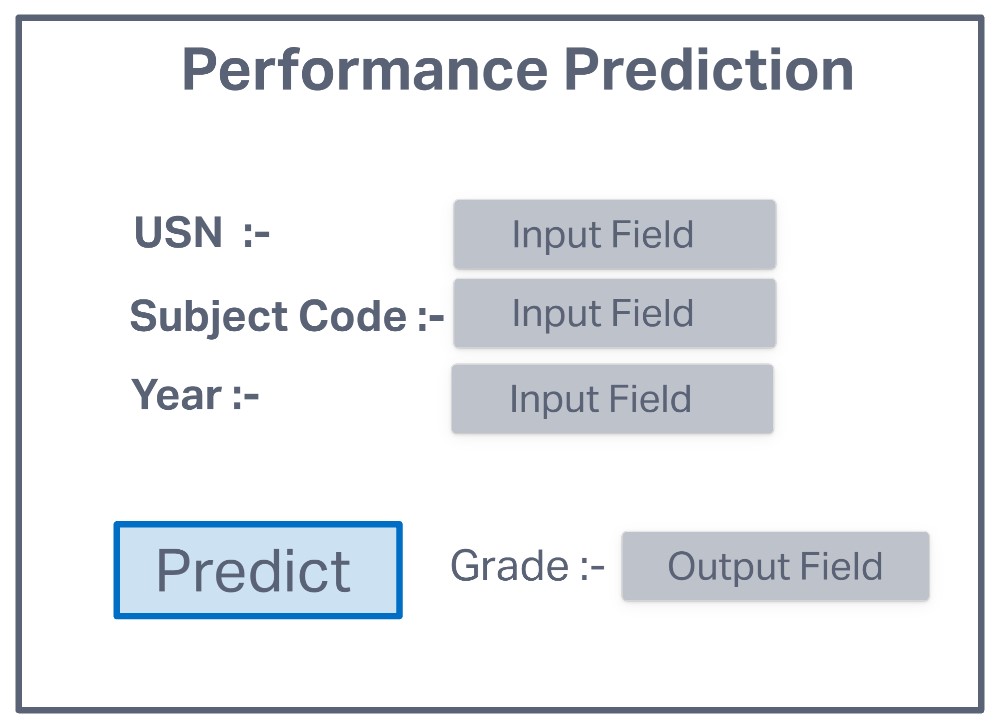
**Wireframe diagrams**

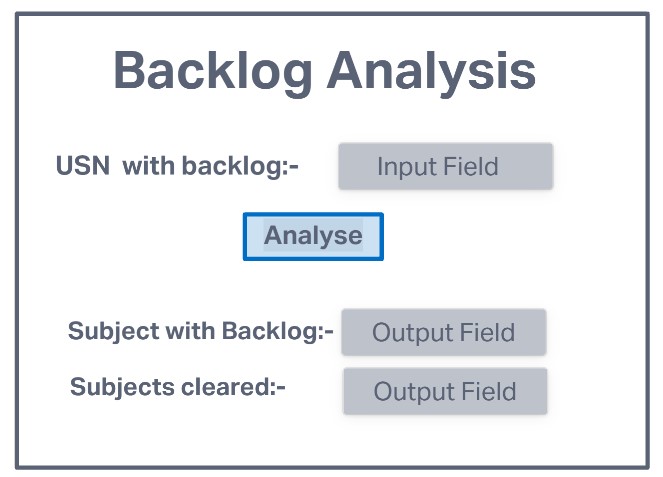
**Wireframe diagrams**

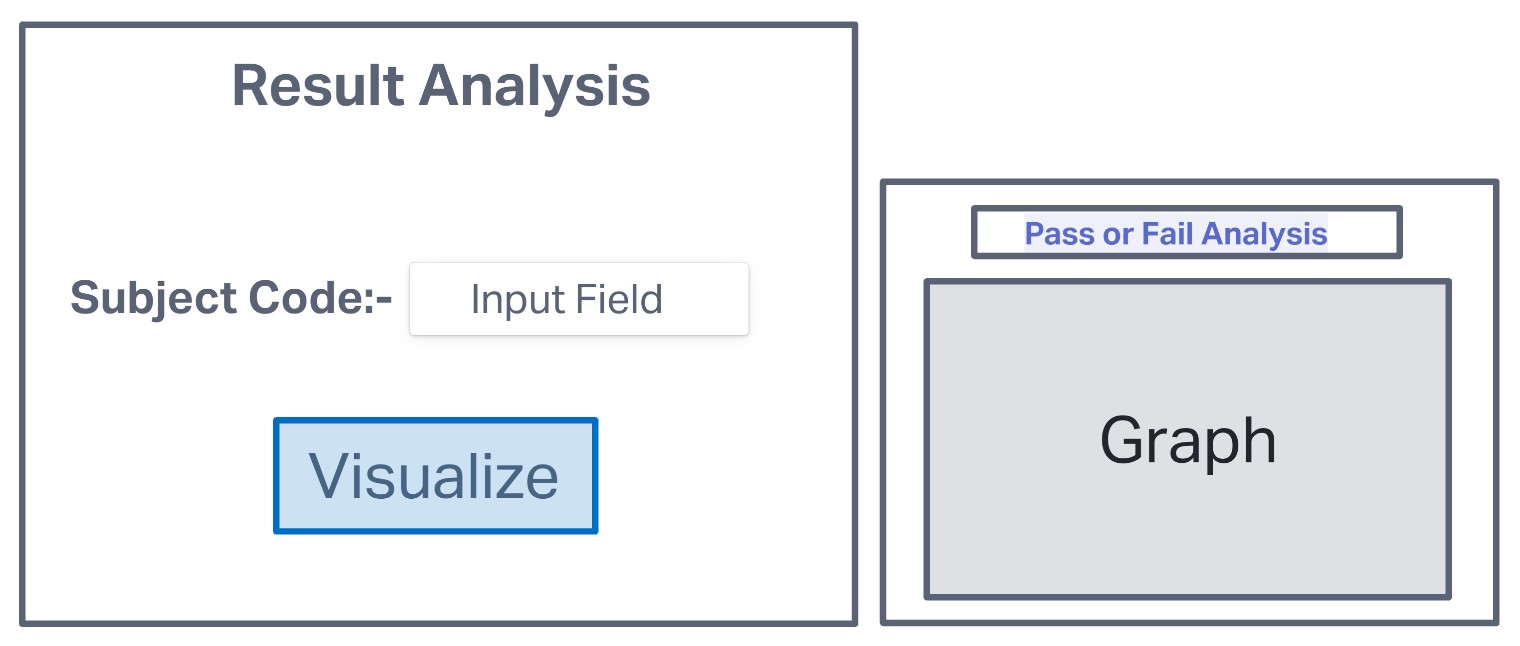


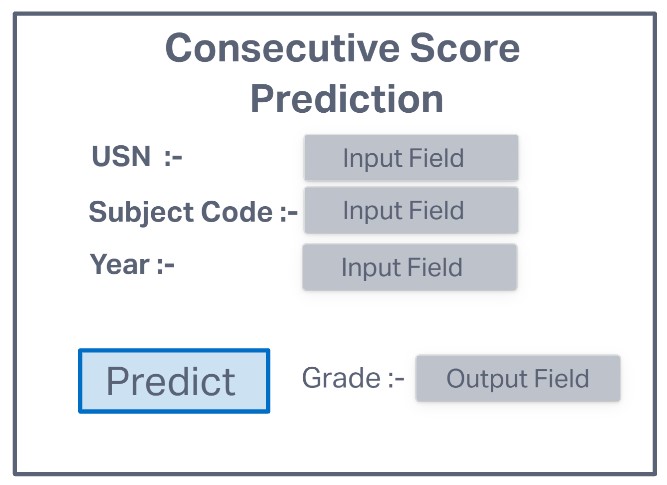
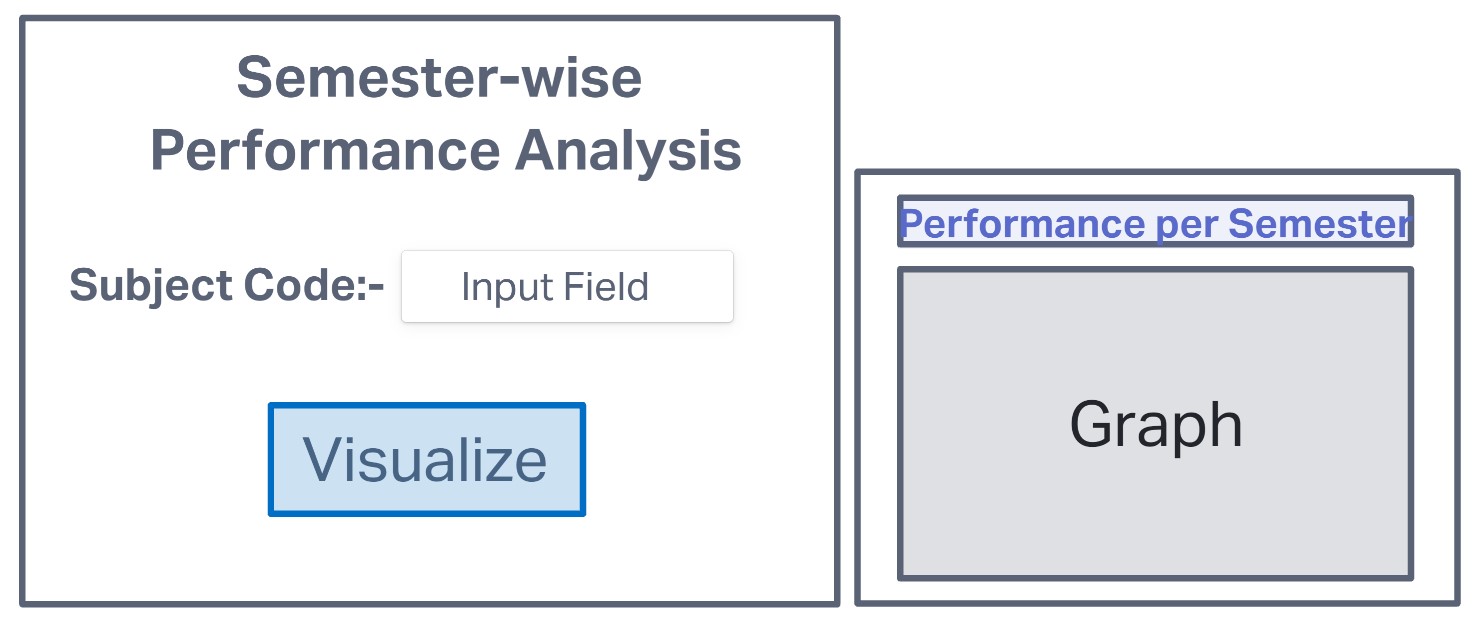
****

****

****

****

****

****

**Design**

**Design**

Desktop application for Result Data analysis is designed as shown in the flowchart below.

When we run the program, the Login page pops up, the user has to provide credentials like Name and Password. Once the credentials are authenticated, the user can then access the apllication to avail its features.

There is a Home page or a Main Menu Page, providing an option for the user to choose among the given objectives of the application.

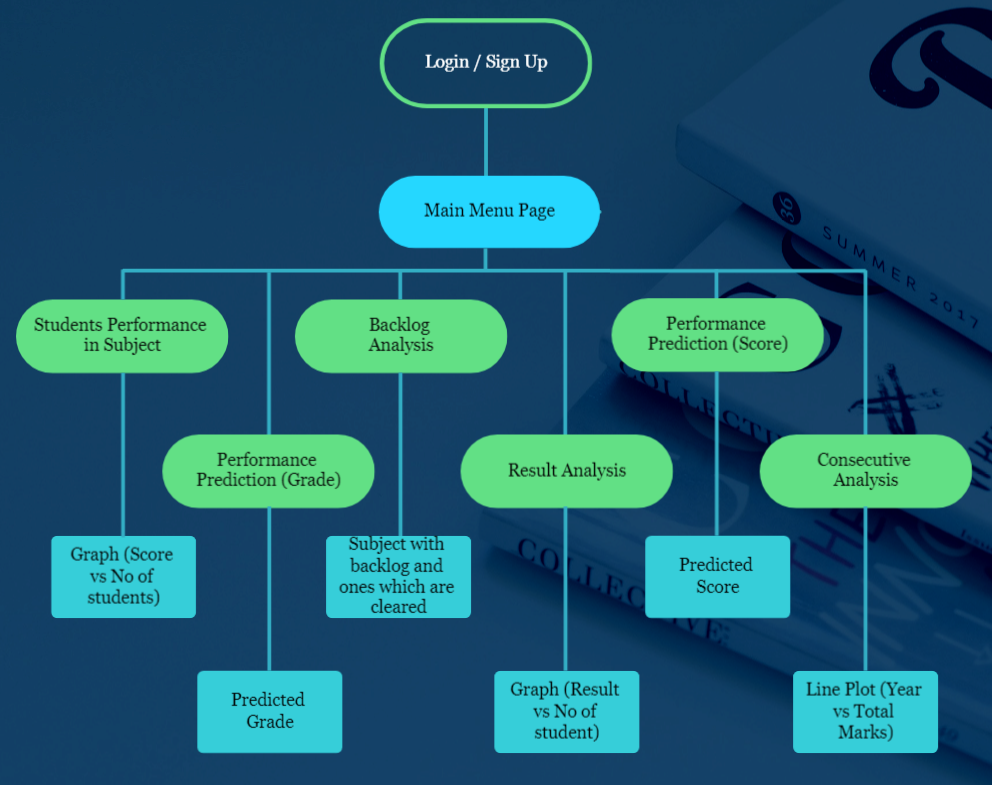
1. Students Performance in Subject - A Bar graph with Score against the number of students attending the subject is visualised.

2. Perfomance Prediction - A prediction of Grade is done and the same is displayed.

3. Backlog Analysis - Subject with backlogs and displayed along with the ones that have been cleared in the successsive attempt.

4. Result analysis - number of sujects passed or failed in a particular subject is visualised.

5. Semesterwise-performance analysis- A Line plot of the Exam and total marks obatined is displayed.

6. Consecutive Score Prediction - A prediction of score in the consecutive examination is made and the same is displayed.

**Development details**

**Development details**

**7.1 Authentication of users**

Our Application is secure and only users with proper authentication can access. The user has to first create an account and then login to it in order to gain access. The information about the users will be stored in a database.

We use SQLlite Database to stored user details. The users have to provide their Name and Password to create and login to thier account.

**7.2 Desktop GUI**

This application has been developed using Tkiner GUI for desktop application. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. The tkinter framework provides Python users with a simple way to create GUI elements using the widgets found in the Tk toolkit. Tk widgets can be used to construct buttons, label, entry, message, menus, data fields, etc. in a Python application.

**7.3 Student perfomance in a subject**

One of the objectives of this application is to visualize the student performance in a particular subject based on the total marks obtained.

In a particular subject the average performance of students is visualized based on the total marks obtained.

A user has to enter the subject code of the desired subject to view the performance of all students in that subject.

A bar graph is depicted for the number of students against total marks obtained in the given subject.

**7.4 Performance Prediction**

The objective is to predict the performance of a particular student based on the subject code. We use Machine Learning or MLib package in Pyspark to solve this. Subject Code, USN of the student and Year of the exam is given as input by the user, the grade of that students in the subject in the next attempt is displayed.

**7.5 Backlog Analysis**

The objective is to analyze whether a student with backlog has cleared the exam in the successive attempt. This analysis is done based on the Examtype specified in the dataset, if the examtype if specified as 'regular' then the subject has been cleared by the student. On providing the USN of a student, the subjects in which the student has a backlog is listed out, and the subjects which have been cleared in the successive attempt will be displayed. If the students has not cleared any backlogs, a message stating the same is displayed. If the student does not have any backlog then a message stating subjects have not been failed will be displayed.

**7.6 Result Analysis**

The objective is to visualize the number of students passed or failed in a particular subject. This analysis is done using the result specified in the dataset which can be either Pass(P) or Fail(F). A Bar graph is plotted using Matplotlib, a python library. The application takes subject code as the input and the graph of that particular subject is shown in a separate window.

**7.7 Semester-wise Performance Analysis**

The objective is to visualize the student performance based on a particular year or semester. The analysis is done by taking average of total marks obtained by a student in a particular semester. A line chart is plotted to visualize the analysis made. The user has to enter the students USN and performance graph of the same will be displayed.

**7.8 Consecutive Score Prediction**

The objective is to predict the score of a student in the consecutive semester based on the score of the similar subject in present or previous semesters. We use Machine Learning or MLib package in Pyspark to solve this. Subject Code, USN of the student and Year of the exam is given as input by the user, the total marks of that students in the specified subject in the next semester is displayed.

**Screenshots**

**Screenshots**

**8.1 Login Page**

The user has to first create an account and then login to it in order to gain access. The information about the users will be stored in a database.

We use SQLlite Database to stored user details. The users have to provide their Name and Password to create and login to thier account.



Figure 9: Login\_Page

**8.2 Home Page**

The home page provides as interface for the user to select the desired function in the application. There are buttons specifying the six objectives of the application.

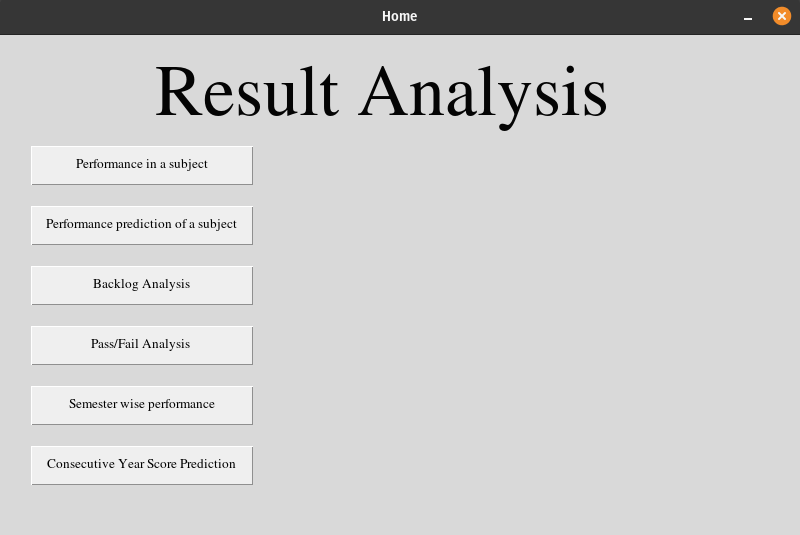


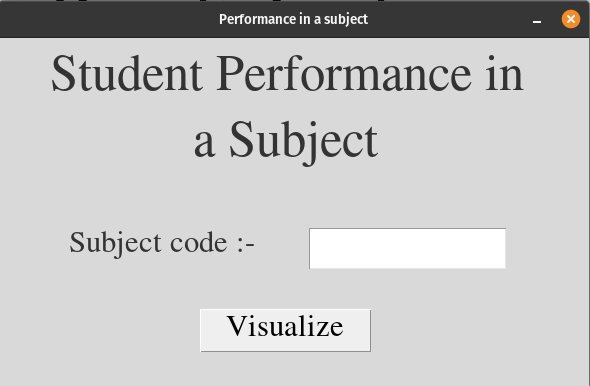
Figure 10: Home Page

**8.3 Student performance in a subject**

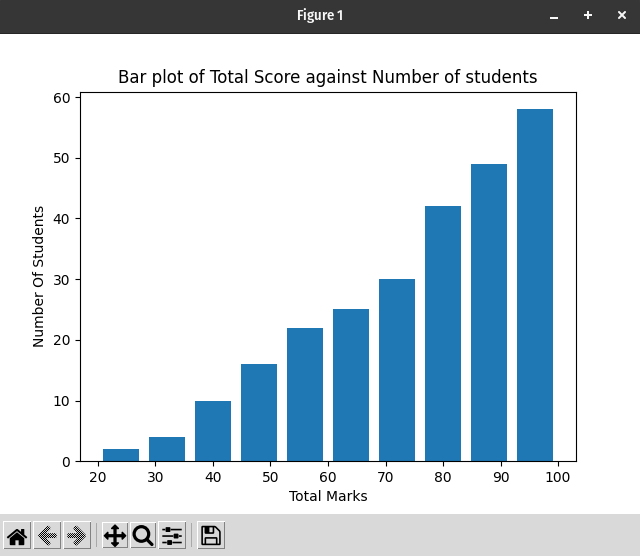
In a particular subject the average performance of students is visualized based on the total marks obtained.

A user has to enter the subject code of the desired subject to view the performance of all students in that subject.

A bar graph is depicted for the number of students against total marks obtained in the given subject.



Sub-Figure a: Input

****

Sub-Figure b: Output

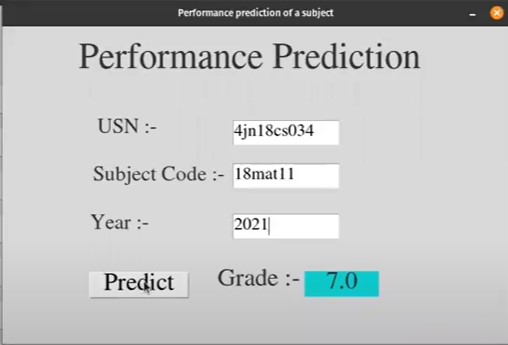
Figure 11: Student performance

**8.4 Performance Prediction**

Subject Code, USN of the student and Year of the exam is given as input by the user, the grade of that students in the subject in the next attempt is displayed.



Sub-Figure a: Input

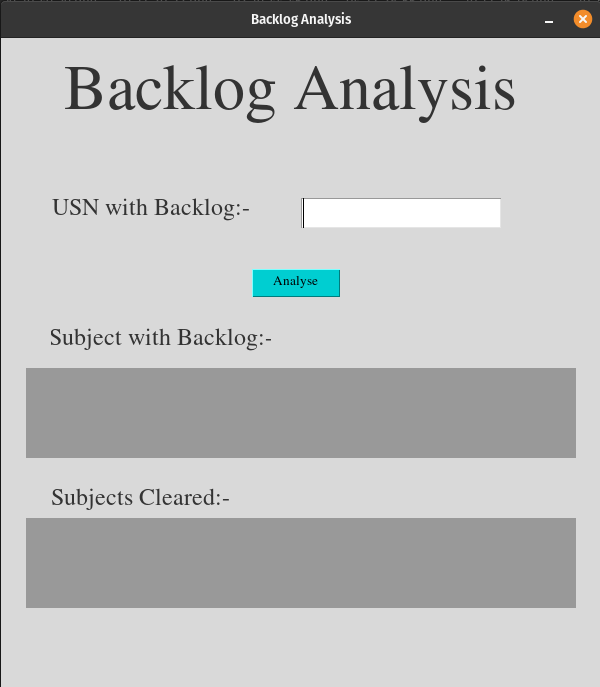
****

Sub-Figure b: output

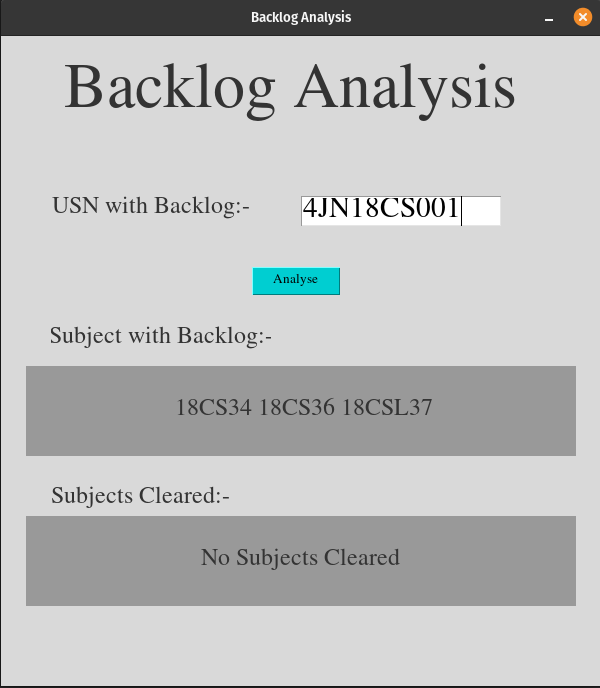
Figure 12: Performance Prediction

**8.5 Backlog Analysis**

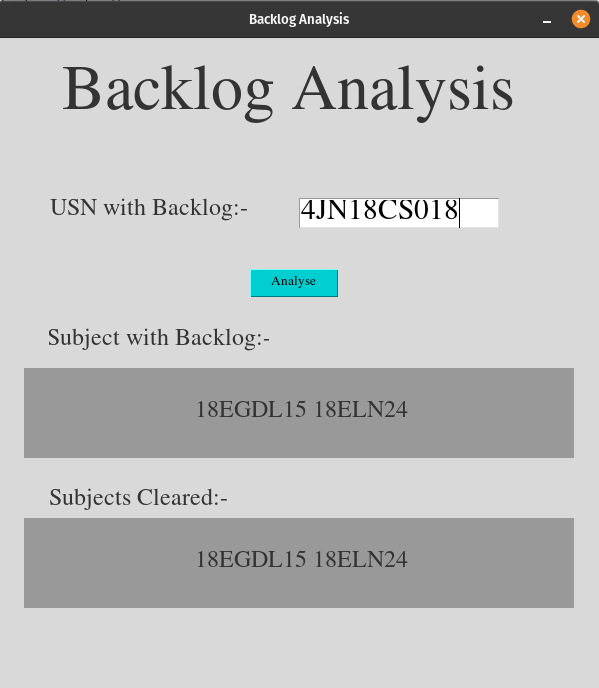
On providing the USN of a student, the subjects in which the student has a backlog is listed out, and the subjects which have been cleared in the successive attempt will be displayed. If the students has not cleared any backlogs, a message stating the same is displayed. If the student does not have any backlog then a message stating subjects have not been failed will be displayed.



Sub-Figure a: Input



Sub-Figure b: Output1

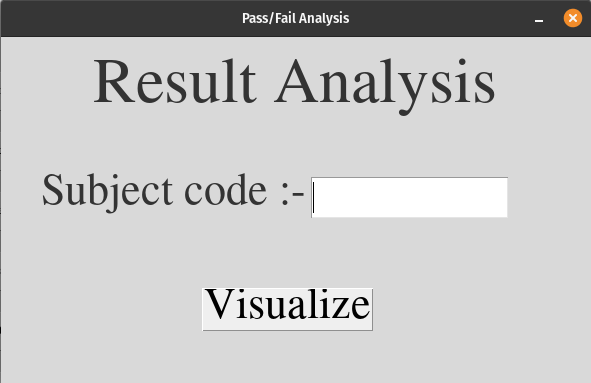


Sub-Figure c: Output2

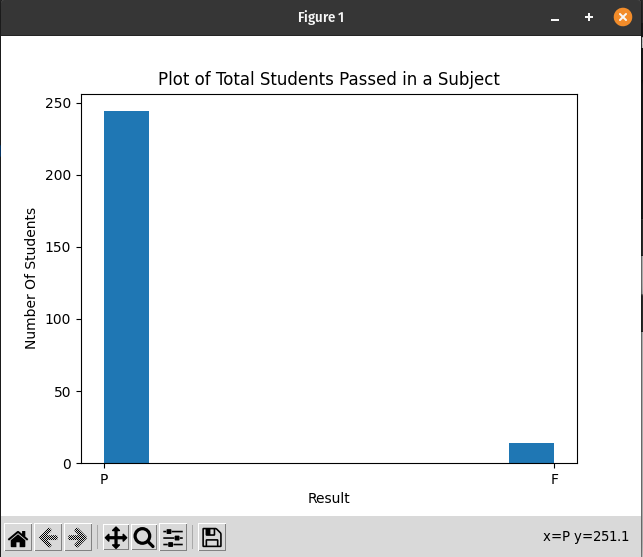
Figure 13: Backlog Analysis

**8.6 Result Analysis**

The application takes subject code as the input and the graph the students passed and failed in that particular subject is shown in a separate window.



Sub-Figure a: Input

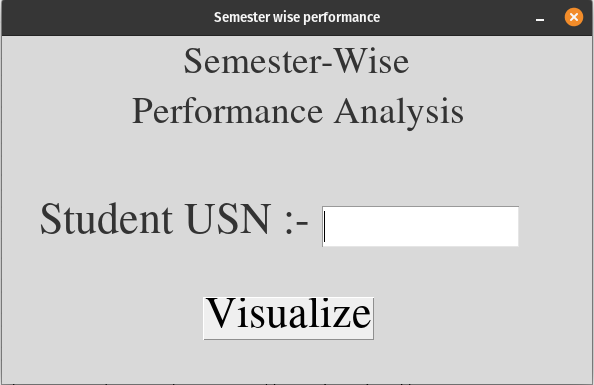
****

Sub-Figure b: Output

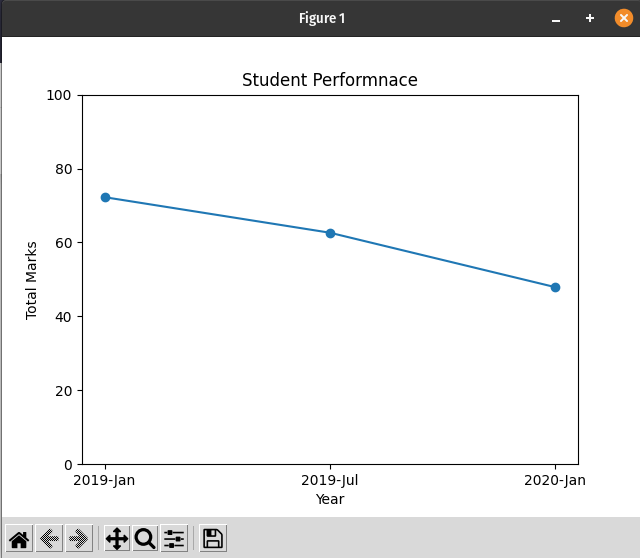
Figure 14: Result Analysis

**8.7 Semester-wise Performance Analysis**

A line chart is plotted to visualize the analysis. The user has to enter the students USN and performance graph of the same will be displayed.

****

Sub-Figure a: Input

****

Sub-Figure b: Output

Figure 15: Semester-wise Performance Analysis

**8.8 Consecutive Score Prediction**

Subject Code, USN of the student and Year of the exam is given as input by the user, the total marks of that students in the specified subject in the next semester is displayed.

****

Sub-Figure a: Input

****

Sub-Figure b: Output

Figure 16: Consecutive Score Prediction

**Conclusion**

**Conclusion**

Successful implementation of Desktop GUI to visualize and Predict the Data from the given Results Dataset. The Objectives recognised are implemented using PySpark in Jupyter Notebook and GoogleColab. After Obtaining succesful results, we integrate the same to create a Desktop application, which is done using python's Tkinter library.

For visualization of data under various constraints and variables, we use Matplotlib plotting library. For prediction of future results or trends in the dataset, we use Machine Learning algorithms like Random Forest and Linear Regression.

A perfectly working desktop application has been developed. This Result analysis application contains buttons for navigating among the six objectives namely, Visualisation of the performance of students in a particular subject based on total marks obtained. Prediction of the performance of a particular student based on the subject code using machine learning. To check whether a Student with a backlog clears the exam in successive attempts. Visualisation of subcode versus number of students who pass or fail that subject. Visualization of student performance based on a particular year. Predicting the score of a student in the consecutive semester based on the score of the similar subject in present or previous semesters.

**References**

**References**

[1] Anand Gupta, Hardeo Kumar Thakur, Ritvik Shrivastava, Pulkit Kumar, Sreyashi Nag. A Big Data Analysis Framework Using Apache Spark and Deep Learning

[2] V Srinivas Jonnalagadda, P Srikanth, Krishnamachari Thumati, Sri Hari Nallamala. A Review Study of Apache Spark in Big Data Processing

[3] Shikha Sonii, Amritpal Singh . Predictive Modelling In IoT Using Apache Spark

[4] D. P. Acharjya, Kauser Ahmed P . A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools

[5] Raswitha Bandi, J Amudhavel, R Karthik. Machine Learning with PySpark - Review

[6] Abdul Ghaffar Shoro, Tariq Rahim Soomro. Big Data Analysis: Apache Spark Perspective

[7] Poonam Salwan, Veerpaul Kaur Maan. Integrating E-Governance with Big Data Analytics using Apache Spark

[8]Jacob Reece, SeongYong Hong. Big data analytics for smart sports using apache spark

[9] Amol Bansod. Efficient Big Data Analysis with Apache Spark in HDFS

[10] Roopa R, Varsha Ryali, Tejasvi Shrivastava, Syed Mahmood Nabeel Anwar. Aadhaar Data Analysis Comparison in MapReduce, Hive and Spark