**SYNOPSIS**

Report on

**STUDENT BEHAVIOUR ANALYSIS**

By

**SNIGDHA PRATAP (202410116100209)**

**TANISHA (202410116100216)**

**SHRUTI SAGAR (202410116100203)**

**TANISHKA SINGH (202410116100219)**

**Session:2025-2026 (III Semester)**

Under the supervision of

**Dr. Neelam Rawat**

**Ms. Shruti Aggarwal**

### KIET Group of Institutions, Delhi-NCR, Ghaziabad



### Department Of Computer Applications

**KIET GROUP OF INSTITUTIONS, DELHI-NCR, GHAZIABAD-201206**

**Student Behaviour Analysis: Nuisance Creation During College Events**

**Abstract**

College events often face disruptions from student nuisance behaviour (e.g. fights, vandalism, crowding) that compromise safety and enjoyment. This project aims to build a data-driven system that uses historical incident records to predict the risk of nuisance at upcoming events. We will develop predictive models (e.g. classification or scoring algorithms) in Python using past data on event type, time, location, and participant behaviour. The outputs will feed into an interactive **Power BI** dashboard for planners to categorize event risk (low/medium/high) before the event. By shifting from reactive security to proactive risk management, the system is expected to help organizers allocate resources (e.g. security personnel) more effectively and reduce disturbances.

**Introduction**

Unruly or nuisance behaviour during college events (festivals, games, concerts, etc.) can endanger students and damage institutional reputation. Defining the problem, we seek to analyze patterns in historical event data (incident logs, student reports) to **forecast where and when disturbances are likely**. The project scope is limited to *pre-event risk prediction* (no real-time surveillance); using only past records of behaviour (e.g. noise complaints, fights, rule violations). By employing data analytics on these records, we aim to predict an event’s likely disturbance level. Objectives include: (1) designing a database to store historical events and incidents, (2) developing an analytics model (Python/Pandas, scikit-learn) to compute risk scores for future events, and (3) building a Power BI dashboard to visualize predicted risk categories and historical trends. A comprehensive view of student activities (attendance, access card swipes, prior incidents, etc.) can yield a “holistic” profile that enhances predictions. This system will support university decision-makers in planning safer events.

**Problem Statement**

* **Issue:** Student nuisance behaviours (e.g. fighting, vandalism, harassment, intoxication) occur unpredictably at campus events. These incidents disrupt the event, risk injuries, and create liability for the college.
* **Who It Affects:** Attendees (students, staff), event organizers, campus security, and the wider community. Disruptions can deter future participation and undermine campus safety.
* **Current Limitations:** Colleges primarily use reactive measures (security guards, CCTV monitoring) and checklists for event safety. There is **no established system that uses data analytics to flag high-risk events in advance**. We lack early-warning tools to anticipate nuisance incidents. Industry insights note that many event planners still operate reactively, whereas modern crowd management is moving toward predictive approaches. Without analytics, organizers cannot easily identify patterns (e.g. certain event types or locations) associated with high nuisance rates. This project addresses that gap by providing a predictive, data-driven pre-event risk assessment tool.

**Literature Review**

Data-driven dashboards and predictive analytics have been used in various domains to improve safety and behaviour monitoring. For example, Wark (2022) demonstrated using **Power BI** in a zoo setting to integrate multiple behavioural data streams and build dashboards that summarize animal welfare and detect changes in behaviour. This shows that business intelligence software can combine data cleaning, analytics, and visualization in a stepwise workflow, aiding proactive management. In education, predictive analytics is increasingly used for student success and risk assessment. Desouza & Smith (2016) discuss how universities collect broad data on student activities to “nudge” behaviour, emphasizing holistic views of students by integrating course performance, dining patterns, gym usage, etc. Data can be mined to detect trends and predict outcomes, enabling interventions. Similarly, specialized dashboards (e.g. SchoolAnalytix) leverage Power BI’s predictive modeling features to forecast student enrollment and identify at-risk learners.

In the context of crowd safety, recent industry analyses emphasize predictive over reactive strategies. One July 2025 event-planning article notes *“emergent technologies are transforming crowd management from reactive to predictive approaches. Advanced data analytics and machine learning models can now anticipate potential crowd-related challenges before they occur”*. Such analytics can simulate crowd behaviour and identify safety vulnerabilities ahead of time. In occupational safety research, big data analytics have achieved high prediction accuracy: for example, a model built from 112 million safety observations predicted worksite incidents with 80–97% accuracy. This indicates strong potential for forecasting incidents when rich historical data exist. Collectively, these studies suggest that applying predictive analytics and dashboards (especially in Power BI) can enhance monitoring and preemptive planning in safety-related settings.

**Proposed System**

The proposed solution consists of **three main components**: historical data input, an analytics model, and a Power BI dashboard.

* **Historical Data Input:** We will compile past records of college events and any documented nuisance incidents. Relevant fields include event ID, date, location, type (e.g. sports, concert), attendee counts, and logged incidents (nature of nuisance, time, involved individuals or groups). This data might come from security logs, police reports on campus, or incident forms. A structured SQL database will store this cleaned historical dataset.
* **Analytics Model:** Using Python, the system will preprocess the data (e.g. one-hot encoding of event types, aggregation of past incident counts) and train a predictive model. Possible approaches include classification or regression (e.g. decision tree, random forest, or logistic regression) to output a **risk score or category (low/medium/high)** for future events. The model will be trained on historical patterns (e.g. high incidence events had X features). Performance will be validated via cross-validation on past data. The prediction module will be limited to offline (pre-event) use – for each upcoming event’s characteristics, the model will output a risk level.
* **Power BI Dashboard:** We will create an interactive dashboard using Microsoft Power BI to visualize the analytics. After predictions are made, the results and historical data will be visualized. Dashboards will show risk categories (e.g. a color-coded list of upcoming events by risk), charts of incident trends (e.g. incidents by event type or time of day), and filters to explore by demographics or venues. For example, Figure 1 shows a **sample behaviour summary dashboard** (adapted from Wark 2022which highlights top behaviours and time distributions. Our dashboard will similarly summarize student nuisance metrics for planners. Power BI’s drag-and-drop interface allows building such dashboards easily, integrating the cleaned data and predictive outputs.

*Figure: Sample Power BI dashboard summarizing key behaviours and trends (from Wark 2022). The proposed system will use similar interactive reports to display predicted risk and historical patterns.*

The system is intended for planning use: organizers input the details of a planned event into the system, the predictive model assigns a risk score based on historical analogs, and the Power BI dashboard presents this along with related charts. No real-time monitoring (e.g. live cameras) is involved – all insights come from historical nuisance behaviour data and static event features.

**Modules Description**

The system comprises the following modules:

* **Data Ingestion & Processing:** Collect historical event and incident logs from university sources. Clean and normalize (e.g. resolve missing values, standardize categories). This module loads data into relational tables (see Database Design below).
* **Feature Engineering:** Transform processed data into model-ready features (e.g. count of past incidents at a venue, time of day, number of attendees).
* **Analytics & Prediction:** Apply statistical or machine-learning algorithms (using Python’s scikit-learn) to learn from historical data and compute risk predictions. Model outputs could be probabilities or discrete risk categories. For instance, the system might learn that concerts after 10pm in certain locations have historically higher nuisance rates, and use that to flag similar events.
* **Scoring Logic:** Convert the model’s output into user-friendly scores or labels (e.g. “High Risk” if predicted probability > 0.7). These risk scores are stored back in the database for reporting.
* **Dashboard Reporting:** Using Power BI Desktop, build reports and visuals that connect to the database. This includes charts (bar graphs, heatmaps) and slicers (filters by date, event type, location). The dashboards refresh from the stored predictions and history. Power BI’s built-in predictive modeling features and KPI definitions can be leveraged here.

Throughout these modules, Python’s data libraries (Pandas for dataframes) prepare data, while Power BI handles visualization. The modules will be orchestrated so that once new event details are entered, the model updates the risk score in the database, and the dashboard automatically reflects the new information.

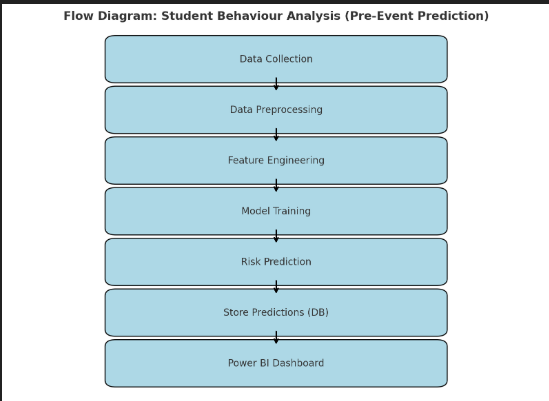
**Database Design**

The core database will be a relational SQL (e.g. MySQL or PostgreSQL) with tables for events, incidents, and related entities:

* **Events:** *EventID (PK), Name, Date, Time, Location, Type, ExpectedAttendance, etc.* Each row represents a planned college event.
* **Students (optional):** *StudentID (PK), Demographics, Enrollment info, etc.* (If modeling individual-level behaviour is needed.)
* **Incidents:** *IncidentID (PK), EventID (FK), StudentID (FK, nullable), Category, Description, Timestamp, Severity.* Each nuisance or rule-violation report linked to an event (and possibly a student). For example, an incident could be “fight” with associated details.
* **Behaviour Types:** *CategoryID (PK), Name, Description.* (E.g. Noise, Harassment, Vandalism.) Incident. Category is a foreign key to this table.
* **Predicted Risk:** *EventID (FK), RiskScore, RiskCategory, PredictionDate.* Stores the output of the analytics model for each upcoming or past event.

Entity relationships: *Events 1–n Incidents*, and *Students 1–n Incidents* (if tracking individual offenders). The Predicted Risk table links to Events via EventID. This design supports queries like “show all incidents for event X” and allows aggregating by event or by behaviour type. Indexing on date and event type will speed up analytics queries.

**Algorithm / Process Flow**

****

The processing workflow proceeds as follows:

1. **Data Collection:** Gather historical data on college events and nuisance incidents (e.g. last 5 years). Load into the SQL database (Events, Incidents tables).
2. **Data Cleaning:** In Python, connect to the database and clean the dataset (handle missing values, normalize categories). For example, ensure event types are consistent and timestamps are correctly formatted.
3. **Feature Engineering:** Derive features such as *incident counts per venue*, *day of week/time slot*, *past incident rates per student*, etc. Join related tables to assemble a machine-learning dataset.
4. **Model Training:** Split historical data into training/testing sets. Use algorithms (e.g. Random Forest classifier) to predict a target label (e.g. HighRisk vs LowRisk) or numeric score. Evaluate accuracy (e.g. via cross-validation).
5. **Model Validation:** Tune parameters to avoid overfitting. Validate that predictions match historical outcomes. (Prior research has shown incident models can reach high accuracy.)
6. **Prediction:** Apply the validated model to features of upcoming/planned events to compute a risk score. Store these results in the Predicted Risk table (with timestamp and category label).
7. **Dashboard Generation:** In Power BI, connect to the SQL database. Import both historical incident data and the new Predicted Risk results. Construct visuals (charts, tables) that show risk categories and key metrics. For instance, the dashboard might highlight events flagged as “High Risk” and display historical incident trends by event type.
8. **Review & Iterate:** Review the dashboard and model performance. If needed, refine features or model (e.g. add new data fields, adjust thresholds) and update the database and dashboard.

All data handling is based on archival (historical) behaviour; there is no live sensor input. This offline pipeline ensures predictions rely solely on past nuisance behaviour patterns.

**Hardware & Software Requirements**

* **Hardware:** A standard PC or server with moderate specifications (e.g. 8+ GB RAM, multi-core CPU, enough disk for database and data processing) should suffice. No specialized hardware (e.g. GPUs) is required for this scale of analytics.
* **Software:**
  + **Operating System:** Windows 10/11 or any recent Linux distribution. Power BI Desktop requires Windows, or use Power BI Service for cloud.
  + **Database:** SQL database engine (e.g. Microsoft SQL Server, PostgreSQL, or MySQL) to store structured data.
  + **Programming:** Python 3.x environment. Key libraries: Pandas for data manipulation, Scikit-learn for machine learning, SQL connector libraries (e.g. SQLAlchemy) for database access.
  + **BI Tool:** Microsoft Power BI Desktop (free version) for dashboard creation; optionally Power BI Pro or Server for sharing reports.
  + **Others:** Git (version control), and Office software for documentation.

This setup allows integration: Python scripts can be run on the same machine or server where the database resides, and Power BI can connect to the SQL database either locally or via network.

**Expected Outcomes**

The system is expected to yield the following outcomes:

* **Risk Categorization:** Each planned event will be automatically classified into risk levels (e.g. “Low”, “Moderate”, “High”). Organizers can immediately see if an event falls into a higher-risk category based on historical patterns.
* **Power BI Insights:** The dashboard will offer interactive insights. For example, it may display a chart of nuisance incident frequency by event type or time of day, highlight top problem behaviours, and filterable lists of upcoming events by risk. These visual insights help planners understand *why* an event is risky.
* **Proactive Planning:** By knowing which events carry elevated risk in advance, administrators can allocate extra security, adjust event timing, or implement preventive measures. Over time, this should reduce the number and severity of actual disturbances. As one analyst notes, integrating predictive models and dashboards helps create “safer, more efficient, and more enjoyable experiences” at large gatherings.
* **Data-Driven Decision-Making:** Replacing guesswork with analytics makes event safety management evidence-based. Stakeholders can track trends (e.g. a decline in incidents after interventions) and continually refine policies.

Overall, we anticipate a measurable reduction in event disturbances as the solution enables timely interventions. The Power BI dashboard makes the predictions transparent and actionable for non-technical users, thereby enhancing campus safety culture.

**Timeline**

A realistic 3-month (≈12-week) schedule is:

* **Weeks 1–2:** Project kickoff; detailed requirements analysis. Gather historical data sources and design the database schema (entities and relationships).
* **Weeks 3–4:** Implement database and ingest historical data. Perform exploratory data analysis in Python; clean and preprocess the dataset. Define features.
* **Weeks 5–6:** Develop and train predictive models using the cleaned data. Iterate model selection (try decision trees, logistic regression, etc.) and evaluate with cross-validation.
* **Weeks 7–8:** Integrate the model’s predictions into the SQL database (create PredictedRisk table). Build initial Power BI dashboard prototype connected to the database.
* **Weeks 9–10:** Refine the dashboard with custom visuals, filters, and drill-downs. Validate model predictions on hold-out events and adjust if necessary. Optimize queries and dashboard refresh.
* **Weeks 11–12:** Conduct final testing of the system (end-to-end). Document methodology and findings. Prepare the project report and any presentation materials. Address any remaining bugs or improvements.

Each milestone will include review meetings with stakeholders to ensure alignment. By the end of month 3, the system should be fully functional and demonstrable, with risk predictions and dashboards validated.