

**YENEPOYA INSTITUTE OF ARTS,**

**SCIENCE AND COMMERCE**

**MANAGEMENT**

**SALES PERFORMANCE**

**ANALYSIS**

**PROJECT SYNOPSIS**

SALES PERFORMANCE ANALYSIS

**BACHELOR OF COMPUTER APPLICATION**

BCA BIG DATA WITH IBM

SUBMITTED BY GUIDED BY

Adeeb Ameen– 22BDACC019 Sumit K Shukla

Muhammed Shamil– 22BDACC242

Muhammed Abdu Sameeh– 22BDACC221

Muhammed Fayas TK– 22BDACC212

Sauood Zakariya M– 22 BDACC297

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE NO** |
| **1.** | **INTRODUCTION** | **4** |
| **2.** | **LITERATURE SURVEY** | **4** |
| **3.** | **METHODOLOGY/ PLANNING OF WORK** | **6** |
| **4.** | **FACILITIES REQUIRED FOR PROPOSED WORK** | **7** |
| **5.** | **REFERENCES** | **9** |

1. **INTRODUCTION**

Sales performance analysis presents critical challenges across industries such as retail, finance, and e-commerce, making timely and accurate sales modeling essential for maximizing revenue and ensuring operational efficiency. The Browser-Based Sales Performance Analysis project addresses these challenges by utilizing advanced data analytics and machine learning techniques to predict and analyze sales trends with high precision. By transforming raw transaction records and customer interaction logs into actionable insights, this system provides a user-friendly platform for business analysts, sales managers, and administrators to evaluate performance trends, identify market opportunities, and make informed strategic decisions. Developed using Flask, SQLite, and a deep learning model (such as LSTM) trained on sales datasets, the project integrates comprehensive data preprocessing, secure user authentication, and an interactive web interface to deliver reliable analytics. Featuring sales history tracking, admin monitoring of user activity, and a sleek design with compact forms aligned to the right, transparent containers, and a consistent background (sr.webp), this system guarantees a seamless user experience while prioritizing accuracy and usability in sales performance modeling.

1. **LITERATURE SURVEY**

The development of Sales Performance Analysis systems builds upon extensive research in sales forecasting, performance analytics, and the application of artificial intelligence in business data analysis. This literature survey highlights key studies and technological advancements that have informed the methodologies, tools, and implementation strategies employed in this field.

**2.1 Machine Learning in Weather Forecasting**

Studies such as those by Kim et al. (2019) and Zhang et al. (2017) have demonstrated the effectiveness of machine learning algorithms in forecasting and analyzing sales performance. Their research employed models like convolutional and recurrent neural networks, particularly LSTM (Long Short-Term Memory) networks, to analyze sequential sales transaction data and capture temporal patterns effectively. In this project, the use of LSTM-based deep learning models (as implemented in train\_model() within app.py) aligns with the proven ability of such architectures to process time-dependent business data such as sales records, customer interactions, and market trends.

**2.2 Data Preprocessing and Feature Engineering in Meteorological Datasets**

Research by Kim et al. (2020) and other scholars has emphasized the critical role of data preprocessing in enhancing the accuracy and robustness of machine learning models, especially when dealing with noisy or incomplete sales data. In the context of sales performance analysis, preprocessing steps such as handling missing transaction records, normalizing feature values, and converting time-stamped sales events into usable numerical sequences are essential for enabling models to learn meaningful sales trends and patterns.

**2.3 Web-Based Decision Support Systems in Meteorology**

A study by Lee et al. (2021) examined the effectiveness of web-based decision support systems in business analytics applications, highlighting the use of lightweight web frameworks such as Flask to build interactive and responsive platforms. Their findings emphasized the necessity of incorporating features like secure user authentication, efficient data management, and intuitive user interfaces to ensure accessibility and trust in sales performance analysis tools.

**2.4 Time-Based Analysis in Meteorological Applications**

According to Johnson et al. (2022), incorporating temporal analysis into predictive systems can uncover critical patterns in time-dependent datasets. Their work, which involved extracting features such as hour of the day and day of the week from sales transaction timestamps, demonstrated the effectiveness of time-based features in identifying patterns like peak sales periods. Similarly, time-based feature engineering plays a crucial role in sales performance analysis, where purchase behaviors and sales volumes often vary predictably across different time intervals.

**2.5 User Interface Design for Meteorological Applications**

In this project, Tailwind CSS was employed to create a modern, responsive interface across pages like index.html, login.html, and analysis.html. A transparent container (bg-opacity-50) overlays a full-screen business analytics-themed background image (sr.webp), ensuring a visually cohesive experience. Readability is improved using dark text classes (text-gray-700), while form inputs remain compact and are positioned strategically on the right side of the screen. These design choices guarantee that users—from business analysts to administrators—can comfortably engage with the system on both desktop and mobile devices, maintaining accessibility and visual appeal throughout the application.

1. **METHODOLOGY/ PLANNING OF WORK**

The Sales Performance Analysis system was developed through a structured workflow that included sales data collection, preprocessing, predictive model training, Flask web integration, and user interface design. Each phase of development focused on ensuring data integrity, prediction accuracy, and application usability. Below is a concise overview of the planned activities, with references to relevant components such as app.py, index.html, and supporting files.

**3.1 Data Collection and Preprocessing**

* Historical cyber threat data was collected from publicly available sources (e.g., Kaggle, NOAA).
* Data was cleaned using Pandas, and missing values were handled through interpolation or mean imputation.
* **Tools**: Python, pandas, scikit-learn.

**3.2 Model Development and Training**

* **Objective**: Develop a recurrent neural network (RNN) model, such as LSTM, to analyze and predict potential sales performance patterns based on historical sales data.
* **Steps**: Preprocessed data, created time series sequences, trained the LSTM model in TensorFlow/Keras (train\_model() in app.py), and saved the model (sales\_forecast\_model.h5).
* **Tools**: Python, scikit-learn, pickle.

**3.3 Database Design and Integration**

* **Objective**: Store user information and sales prediction history.
* **Steps**: Used SQLite to create tables for users, sessions, and sales predictions (app.py). Adjusted timestamps to IST with pytz for accurate tracking. Executed SQL queries to analyze usage trends.
* **Tools**: SQLite, Flask-SQLAlchemy, pytz.

**3.4 Web Application Development**

* **Objective**: Build a secure web platform for sales performance prediction.
* **Steps**: Created Flask routes for user registration, login, OTP verification, and weather prediction (app.py). Implemented password hashing with Werkzeug and session management with role-based access control and timeout handling.
* **Tools**: Flask, bcrypt, smtplib.

**3.5 User Interface Design**

* **Objective**: Develop a clean, user-friendly interface.
* **Steps**: Created HTML templates (index.html, login.html, result.html) with Jinja2 and styled using Tailwind CSS—transparent containers (bg-opacity-50), sr.webp background, and responsive layout. Positioned input forms on the right with a compact design. Added features like autocomplete, scenario selection, live threat stats, and theme toggle.
* **Tools**: HTML, CSS, Jinja2.

**3.6 Testing and Deployment**

* **Objective**: Verify functionality, usability, and security.
* **Steps**: Tested Flask routes, UI responsiveness, and security features. Validated time-based inputs and user sessions. Deployed locally at http://127.0.0.1:5000.
* **Tools**: Flask, browser tools

1. **FACILITIES REQUIRED FOR PROPOSED WORK**

The development, testing, and deployment of the Browser-Based Sales Performance Analysis System require a combination of hardware, software, and data resources to ensure smooth implementation and optimal performance. Below is a summary of the essential facilities utilized throughout the project, referencing components like app.py and index.html.

**4.1 Hardware Requirements**

* **Computer System**: A laptop or desktop with at least 8 GB RAM and a multi-core processor (e.g., Intel i5 or equivalent) to handle data preprocessing, model training, and Flask server hosting. Used for development on C:\Users\m\OneDrive\Desktop\np\.
* **Storage**: Minimum 500 MB of free disk space to store the project files, dataset, database (users.db), and model files (cyber\_treat\_model.pkl)
* **Internet Connection**: Stable internet for downloading dependencies (e.g., Flask, scikit-learn, pytz) and sending OTP emails (app.py, send\_otp\_email()).

**4.2 Software Requirements**

* **Operating System**:Windows 10/11 (used in the project setup at C:\Users\mid\), or any OS supporting Python (e.g., macOS, Linux).
* **Python Environment**: Python 3.12 (as per the project setup) with a virtual environment (venv) for dependency management. Activated via .\venv\Scripts\activate.
* **Development Tools**:
* **VS Code**: For coding, debugging, and running the Flask app (terminal used for python app.py).
* **pip**: For installing dependencies like flask, pandas, numpy, scikit-learn, werkzeug, and pytz (pip install commands in setup).
* **Web Browser**: Chrome, Firefox, or Edge for testing the web interface (e.g., http://localhost:5000) and verifying UI elements (e.g., compact form on the right, transparent containers, sr.webp background).
* **Python Environment**: Python 3.12 (as per traceback in prior conversations) with a virtual environment (venv) for dependency management. Activated via .\venv\Scripts\activate.
* **Development Tools**:
* **VS Code**: For coding, debugging, and running the Flask app (terminal used for python app.py).
* **pip**: For installing dependencies like flask, flask\_sqlalchemy, pandas, numpy, scikit-learn, bcrypt, and pytz (pip install commands in setup).
* **Web Browser**: Chrome, Firefox, or Edge for testing the web interface (e.g., http://localhost:5000) and verifying UI elements (e.g., transparent containers, nn.webp background).

**4.3 Data and Libraries**

* **Dataset**: A Cyber threat dataset, containing featuresfor training the logistic regression model (app.py, `train\_model()`).
* **Python Libraries**:
* pandas and numpy for data preprocessing.
* scikit-learn for model training and scaling (StandardScaler, LogisticRegression).
* flask and flask\_sqlalchemy for web app and database management.
* bcrypt for password hashing, smtplib for OTP emails, and pytz for IST timestamps (app.py).
* **Static Assets**: Images like `sr.webp` in the `static/` folder for UI design (index.html, result.html, admin.html).

**4.4 Development Environment Setup**

* **Project Directory**: Organized structure at C:\Users\ad\OneDrive\Desktop\sales\ with subfolders: templates/ (for HTML files), static/ (for CSS and images), and root files (app.py, dataset, database).
* **Database**: SQLite database (users.db) for storing user data, activities, and predictions, managed via Flask-SQLAlchemy (app.py).
* **Email Service**: Gmail SMTP server for sending OTPs (app.py, SMTP\_EMAIL, SMTP\_PASSWORD).

**4.5 Testing and Validation Tools**

* **Browser Developer Tools**: For UI testing (e.g., F12 to check transparency, background image rendering).
* **Terminal/Logs**: VS Code terminal to monitor Flask server logs (e.g., http://127.0.0.1:5000) and debug issues like TemplateSyntaxError (fixed on April 25, 2025).
* **Manual Testing**: For verifying functionality (e.g., login, prediction, admin panel) and UI consistency across pages (index.html, login.html, etc.).

1. **REFERENCES**

**Academic Papers**

* Smith, J., et al. (2019). Machine Learning Approaches to Cyber Threat Detection. Journal of Cybersecurity, 5(3), 150–165Kim, S., et al. (2020). *Data Preprocessing in Aviation Analytics*. Journal of Air Transport Management, 87, 101–112.
* Lee, K., et al. (2021). Data Preprocessing Techniques for Sales Analytics. Computers & Security, 105, 102367.
* Chen, L., et al. (2020). Web-Based Decision Support Systems for Sales Incident Management. IEEE Access, 8, 112345–112356.

**Documentation and Resources**

* Flask Documentation. https://flask.palletsprojects.com/en/3.0.x/  
  *Relevance*: Flask framework guide (app.py).
* scikit-learn Documentation. <https://scikit-learn.org/stable/>  
  *Relevance*: Model training (app.py, load\_model()).