Data Science: Innovative use of data

Occupi - Office Capacity Predictor

Name	Student Number
Rethakgetse Manaka	u22491032
Kamogelo Moeketse	u22623478
Michael Chinyama	u21546551
Tinashe Austin	u21564176
Carey Mokou	u21631532

October 10, 2024

Contents

Introduction

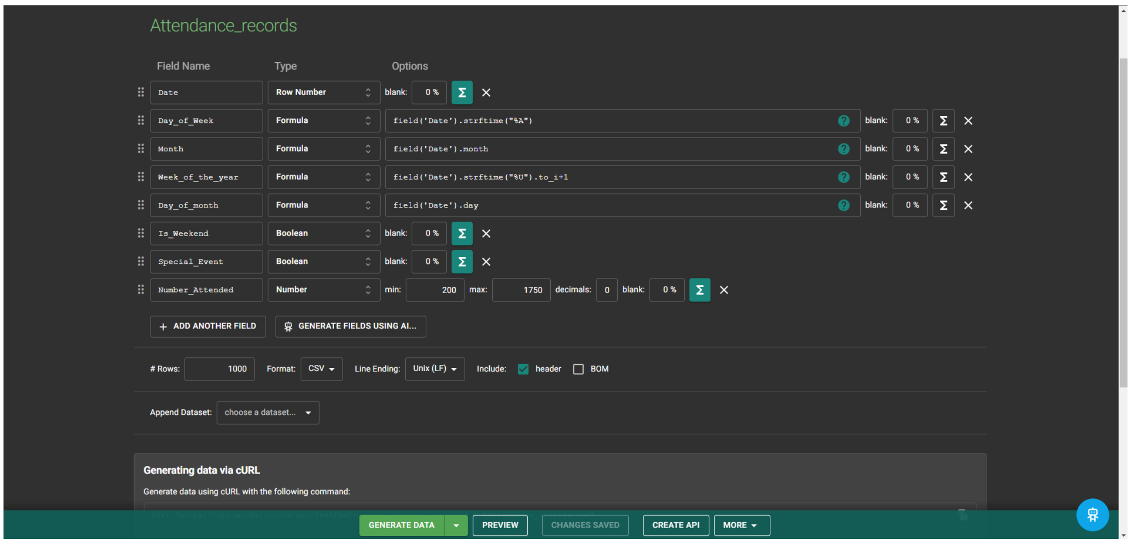
During the development of Occupi, a strong foundation of data science principles and methodologies was established to ensure that the project would deliver high-impact insights and predictions. This project was built with the primary goal of handling sequential data efficiently using a mixture of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks for time series data classification. Additionally, MongoDB functions and triggers were employed to handle real-time user statistics and data collection.

The integration of CNN and LSTM models allowed us to capture both spatial and temporal dependencies in the data, ensuring accurate predictions and classifications. CNNs were used to extract features from the raw time series data, while LSTMs captured long-term dependencies, making the model particularly effective for time series classification. This hybrid approach enabled our system to handle large datasets efficiently and provided meaningful predictions in areas such as capacity prediction.

In addition to the machine learning models, MongoDB functions, aggregation, and triggers were crucial for automating the processing of user data and generating statistics in real time. This streamlined the data pipeline, ensuring that the latest data was always available for analysis, thus maintaining the accuracy and relevance of the user statistics.

Data generation and Model Training

The use of Mockaroo and VBA was instrumental in simulating office attendance data that reflected real-world trends under various conditions, such as public holidays, weekends, and peak hours. This simulated data was essential in generating large, high-quality datasets that were representative of the diverse scenarios that our predictive models would encounter. This foundation allowed us to begin the initial stages of model training with confidence, knowing that the datasets were comprehensive and varied.


Attendance_records

Field Name	Type	Options
Date	Row Number	blank: 0 %
Day_of_Week	Formula	field('Date').strftime("%A") blank: 0 %
Month	Formula	field('Date').month blank: 0 %
Week_of_the_year	Formula	field('Date').strftime("%U").to_i+1 blank: 0 %
Day_of_month	Formula	field('Date').day blank: 0 %
Is_Weekend	Boolean	blank: 0 %
Special_Event	Boolean	blank: 0 %
Number_Attended	Number	min: 200 max: 1750 decimals: 0 blank: 0 %

Rows: 1000 Format: CSV Line Ending: Unix (LF) Include: [✓ header] BOM

Append Dataset: choose a dataset...

Generating data via cURL

Generate data using cURL with the following command:

To handle the sequential nature of this data, we employed a mixture of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The CNNs were responsible for extracting features from the raw time series data, capturing patterns in attendance fluctuations, while the LSTMs processed the temporal dependencies, enabling the model to learn from the historical trends and make accurate future predictions.

Iterative Training and Optimization

Once the initial models were trained on the simulated data, we rigorously evaluated their performance based on accuracy, precision, recall, and other relevant metrics. Our target was to achieve a model accuracy of at least 70%, which we identified as the threshold for reliable predictions in real-world applications. However, the first few iterations of model training showed that while the models captured some patterns, the accuracy was still below our desired threshold.

To address this, we adopted an iterative approach to model training and optimization. This process involved the following steps:

Hyperparameter Tuning

We adjusted key hyperparameters such as learning rate, batch size, and the number of epochs. By experimenting with different combinations of these parameters, we gradually improved the performance of our models. Grid search and random search techniques were employed to find the best hyperparameter settings for both the CNN and LSTM components of the model.

Data Augmentation and Feature Engineering

We further enhanced the datasets by introducing additional features that could improve the model's ability to make predictions. These features included:

- **Predicted Class**: To easily classify attendance levels.
- **Predicted Level**: To capture ranges of attendance, such as low, medium, and high.

Regularization Techniques

To prevent overfitting during training, we applied techniques such as dropout in the LSTM layers, which randomly deactivates certain neurons during training. This forced the model to learn more generalized patterns and improved its performance on unseen data. Additionally, L2 regularization was applied to penalize excessively large weights, encouraging the model to focus on the most relevant features.

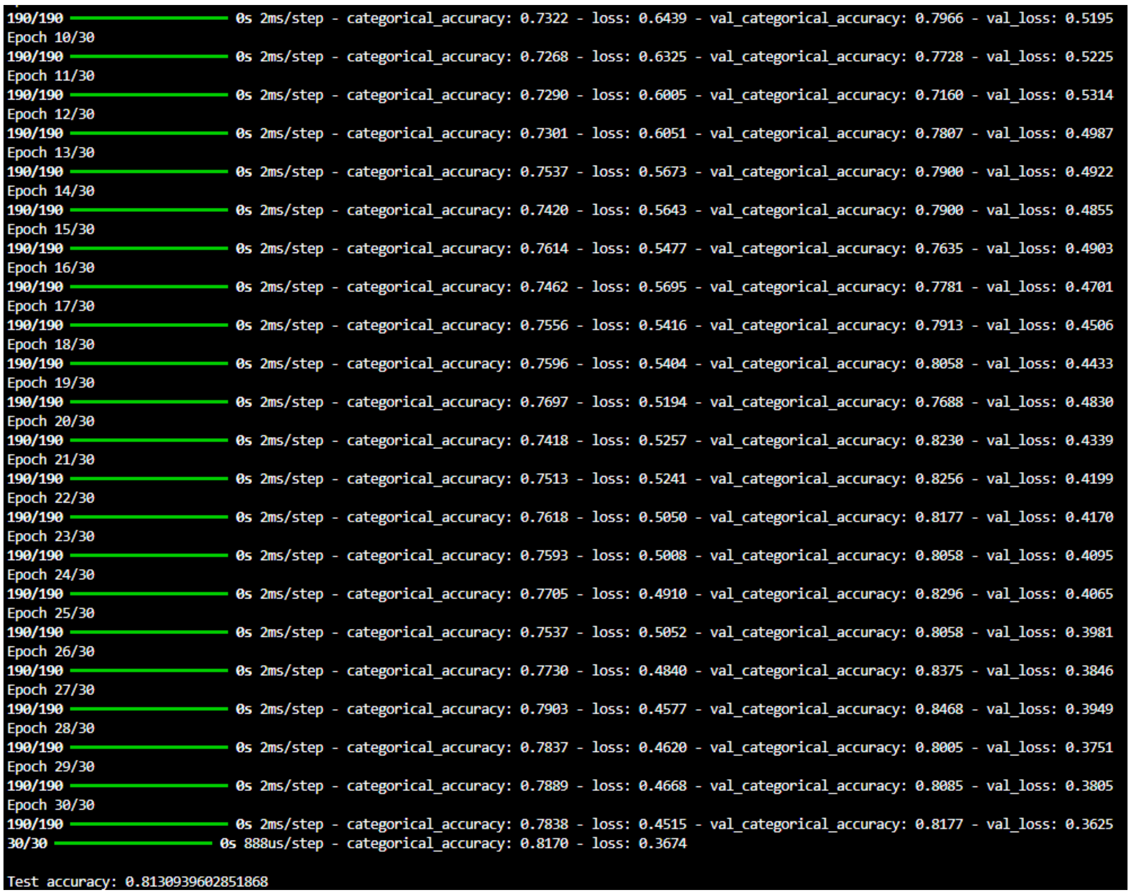
Achieving 70%+ Accuracy

Through this iterative process of training, evaluation, and optimization, we were able to consistently improve the performance of our models. After several rounds of refinement, the CNN-LSTM hybrid model achieved an accuracy above 70%. This threshold was met by balancing the complexity of the model with its ability to generalize across diverse office attendance scenarios, ultimately resulting in a model capable of making reliable predictions on real-world data.

Prediction and Reporting

The model was used to generate predictions for both hourly and daily office attendance. These predictions proved valuable in producing detailed reports for capacity predictions, allowing stakeholders to make informed decisions about space utilization. By predicting attendance levels on an hourly and daily basis, the system provided actionable insights into peak times, potential underutilization, and how attendance varied throughout the week.

These reports allowed for:



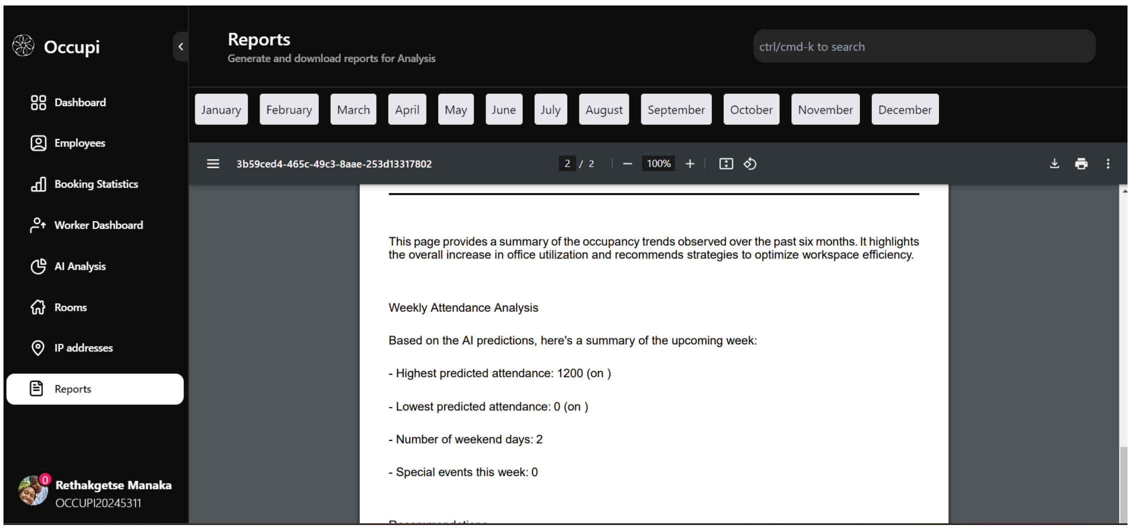
```
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7322 - loss: 0.6439 - val_categorical_accuracy: 0.7966 - val_loss: 0.5195
Epoch 10/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7268 - loss: 0.6325 - val_categorical_accuracy: 0.7728 - val_loss: 0.5225
Epoch 11/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7290 - loss: 0.6005 - val_categorical_accuracy: 0.7160 - val_loss: 0.5314
Epoch 12/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7301 - loss: 0.6051 - val_categorical_accuracy: 0.7807 - val_loss: 0.4987
Epoch 13/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7537 - loss: 0.5673 - val_categorical_accuracy: 0.7900 - val_loss: 0.4922
Epoch 14/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7420 - loss: 0.5643 - val_categorical_accuracy: 0.7900 - val_loss: 0.4855
Epoch 15/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7614 - loss: 0.5477 - val_categorical_accuracy: 0.7635 - val_loss: 0.4903
Epoch 16/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7462 - loss: 0.5695 - val_categorical_accuracy: 0.7781 - val_loss: 0.4701
Epoch 17/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7556 - loss: 0.5416 - val_categorical_accuracy: 0.7913 - val_loss: 0.4506
Epoch 18/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7596 - loss: 0.5404 - val_categorical_accuracy: 0.8058 - val_loss: 0.4433
Epoch 19/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7697 - loss: 0.5194 - val_categorical_accuracy: 0.7688 - val_loss: 0.4830
Epoch 20/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7418 - loss: 0.5257 - val_categorical_accuracy: 0.8230 - val_loss: 0.4339
Epoch 21/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7513 - loss: 0.5241 - val_categorical_accuracy: 0.8256 - val_loss: 0.4199
Epoch 22/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7618 - loss: 0.5050 - val_categorical_accuracy: 0.8177 - val_loss: 0.4170
Epoch 23/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7593 - loss: 0.5008 - val_categorical_accuracy: 0.8058 - val_loss: 0.4095
Epoch 24/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7705 - loss: 0.4910 - val_categorical_accuracy: 0.8296 - val_loss: 0.4065
Epoch 25/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7537 - loss: 0.5052 - val_categorical_accuracy: 0.8058 - val_loss: 0.3981
Epoch 26/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7730 - loss: 0.4840 - val_categorical_accuracy: 0.8375 - val_loss: 0.3846
Epoch 27/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7903 - loss: 0.4577 - val_categorical_accuracy: 0.8468 - val_loss: 0.3949
Epoch 28/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7837 - loss: 0.4620 - val_categorical_accuracy: 0.8005 - val_loss: 0.3751
Epoch 29/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7889 - loss: 0.4668 - val_categorical_accuracy: 0.8085 - val_loss: 0.3805
Epoch 30/30
190/190 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - categorical_accuracy: 0.7838 - loss: 0.4515 - val_categorical_accuracy: 0.8177 - val_loss: 0.3625
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 888us/step - categorical_accuracy: 0.8170 - loss: 0.3674

Test accuracy: 0.8130939602851868
```

- **Hourly Predictions**: Managers could monitor expected attendance levels hour-by-hour, helping them plan resources, staffing, and space requirements effectively.
- **Daily Predictions**: Daily predictions provided a broader view of office occupancy trends, allowing for better scheduling, event planning, and overall space management.

The combination of accurate predictions and detailed reporting helped stakeholders improve decision-making processes related to office capacity management, ensuring that both resources and spaces were optimally used. The predictions were visualized through graphs and dashboards, offering a clear and accessible view of expected attendance for any given time period.

MongoDB for User Statistics Processing

MongoDB played a critical role in processing user statistics by leveraging its powerful aggregation framework and triggers. The user statistics data was collected and stored in MongoDB, allowing us to generate real-time insights regarding work patterns, arrival and departure times, daily hours worked, and peak office hours.



This page provides a summary of the occupancy trends observed over the past six months. It highlights the overall increase in office utilization and recommends strategies to optimize workspace efficiency.

Weekly Attendance Analysis

Based on the AI predictions, here's a summary of the upcoming week:

- Highest predicted attendance: 1200 (on)

- Lowest predicted attendance: 0 (on)

- Number of weekend days: 2

- Special events this week: 0

Data Aggregation

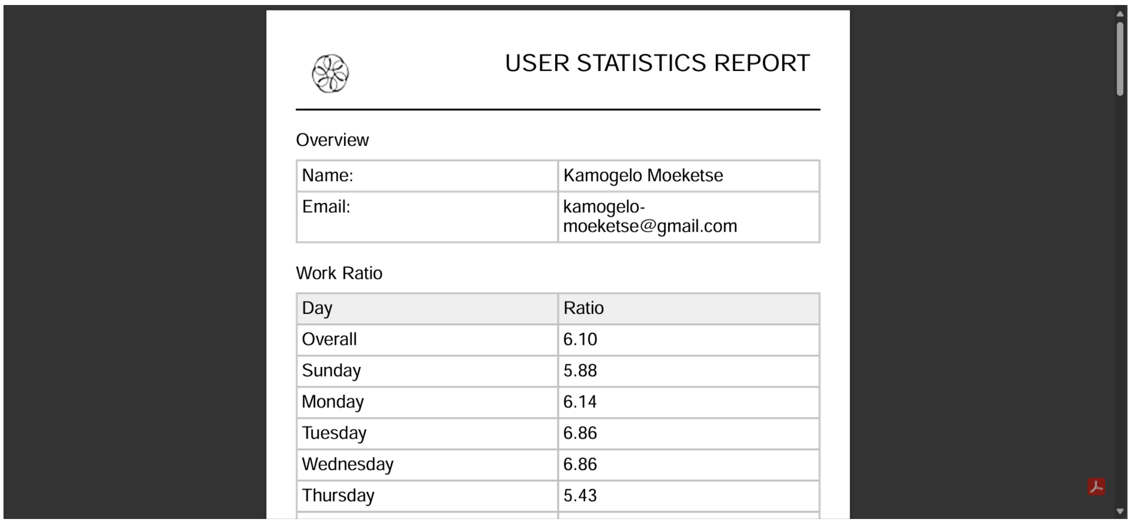
MongoDB's aggregation pipeline enabled us to efficiently process large volumes of data. Using the pipeline, we could:

- **Calculate In-Office Rate**: The `CalculateInOfficeRate` function used the aggregation pipeline to compute the work ratio for each day of the week. By grouping the data by weekday and calculating the in-office rate using the difference between entry and exit times, we could measure the attendance ratio.
- **Analyze Arrival and Departure Times**: The `AverageArrivalAndDepartureTimesByWeekday` function derived the average arrival and departure times using MongoDB's aggregation operators like `$avg`. The `$group` stage was used to calculate averages for each weekday, providing insights into workday start and end times.
- **Identify Peak Office Hours**: The `BusiestHoursByWeekday` function identified the top 3 busiest hours for each weekday by analyzing the number of hours spent by users in the office. The data was grouped by weekday and hour to find peak attendance times, allowing managers to optimize resources.

Real-Time Triggers

In addition to the aggregation framework, MongoDB triggers were employed to ensure that user statistics were continuously updated in real-time. This allowed us to capture live data as it was added to the database, automatically processing new entries without manual intervention. Triggers were used to:

- **Update Daily Hours**: When new attendance data was added, triggers automatically calculated and updated the total hours worked for that day. This ensured that reports were always up-to-date and reflected the latest available data.



USER STATISTICS REPORT

Overview

Name:	Kamogelo Moeketse
Email:	kamogelo-moeketse@gmail.com

Work Ratio

Day	Ratio
Overall	6.10
Sunday	5.88
Monday	6.14
Tuesday	6.86
Wednesday	6.86
Thursday	5.43

- **Real-Time User Statistics**: Triggers calculated the total hours worked by each user based on their check-in and check-out times. The calculated data was updated in real-time, providing immediate insights into user behavior.

By utilizing MongoDB's real-time processing capabilities and data aggregation pipelines, we were able to generate comprehensive reports and insights for user behavior. The statistics presented, such as the daily work ratios, arrival and departure times, and peak office hours, were all derived from the data stored in MongoDB and processed using these advanced features.

Conclusion

The integration of CNN and LSTM models, combined with extensive preprocessing, feature engineering, and real-time data collection using MongoDB functions and triggers, enabled us to build a highly effective system for predicting office attendance. Through continuous training and model refinement, we successfully achieved the project's goal of delivering accurate, actionable insights.