Project Proposal
Analyzing Traffic Occupancy Rate Patterns on San Francisco Freeways

Group 2: Divya Varshini Suravaram (017506580), Soumith Reddy Podduturi (016706612), Veera Vivek Telagani (017503915)

GitHub: https://github.com/veeravivekt/CMPE255-traffic-occupancy-patterns

1. Introduction

1.1 Motivation

This project focuses on analyzing occupancy data to understand usage patterns within a designated space throughout the week. From the data collected from various sensors, we will identify average occupancy rates for each day, revealing peak times during weekdays and a balanced usage pattern on weekends. We will also analyze hourly occupancy data to pinpoint rush hours and detect any trends in daily traffic flow. The insights gained from this analysis will be visualized to help examine decisions for optimizing space management.

1.2 Objectives

This project aims to analyze daily and hourly occupancy patterns to understand traffic flow trends. Specifically, we seek to determine average occupancy rates for each day of the week, highlight peak traffic times during weekdays, and identify balanced usage patterns on weekends. Additionally, we will explore hourly trends to pinpoint rush hours and detect anomalies or consistent flows. The results will be visualized to provide clear and actionable insights, helping inform decisions on optimizing freeway space and managing traffic more effectively.

1.3 Literature Review

Previous research has demonstrated the importance of analyzing high-resolution sensor data for understanding real-time traffic conditions and managing congestion. Studies have shown that weekdays typically exhibit pronounced morning and evening peaks, while weekends tend to have more consistent usage patterns. Data-driven decision-making is increasingly recognized as a powerful tool in infrastructure planning, with visualization playing a key role in translating complex datasets into practical insights.

2. System Design & Implementation Details

2.1 Algorithms Selected

- **Random Forest Classifier (RF):** A Random Forest is an ensemble learning technique that predicts outcomes by mixing numerous decision trees. It works by aggregating individual trees' outputs, either through voting (classification) or averaging (regression). Compared to single decision trees, this reduces overfitting while increasing accuracy.
- **Gradient Boosting Classifier (GB):** Gradient Boosting builds models sequentially, with each model correcting the errors of its predecessor. It combines weak learners (usually decision trees) into a strong learner by minimizing a loss function.
- **Voting Classifier:** The Voting Classifier improves overall performance by combining predictions from multiple models (in this case, Random Forest and Gradient Boosting). A soft voting approach is used, which averages the predicted probabilities of each model, with weights of 2:1 favoring the Random Forest model.
- **Standard Scaler**: It is a data preprocessing tool that standardizes features by removing their mean and scaling to unit variance. This ensures all features are on the same scale and contribute equally to the model's predictions.

Model	**Parameters**
Random Forest	n_estimators=2000, max_depth=None, min_samples_split=2, min_samples_leaf=1, max_features='sqrt', bootstrap=True, class_weight='balanced_subsample', random_state=42, n_jobs=-1
Gradient Boosting	n_estimators=1000, learning_rate=0.005, max_depth=10, min_samples_split=4, min_samples_leaf=2, subsample=0.8, random_state=42
Voting Classifier	n_estimators=1000, learning_rate=0.005, max_depth=10, min_samples_split=4, min_samples_leaf=2, subsample=0.8, random_state=42
Standard Scaler	N/A
Pipeline	scaler=StandardScaler(), ensemble=voting_clf

2.2 Technologies & Tools Used

- **NumPy:** For numerical computations and efficient handling of multi-dimensional arrays.
- **Matplotlib**: For creating static and interactive visualizations.
- **Seaborn**: For high-level, aesthetically pleasing data visualizations.
- **Scikit-learn**: Machine learning models are implemented using tools for preprocessing and evaluating data.
- **Statsmodels**: For statistical modeling and time-series decomposition.
- **SciPy**: For advanced scientific and statistical computations.

2.3 System Architecture

Input Layer: Accepts pre-processed training and testing datasets with labels.
Preprocessing Layer: Scales features using StandardScaler for consistent input to models.
Model Layer:

- **Random Forest:** Ensemble model with 2000 trees, handling class imbalances.
- **Gradient Boosting:** Sequential model optimized for complex patterns.
- **Voting Classifier:** Combines Random Forest and Gradient Boosting with weighted soft voting (2:1).

Pipeline: Integrates preprocessing and ensemble model for streamlined prediction.
Evaluation Layer: Computes accuracy and generates a classification report.

Input Data → StandardScaler → Voting Classifier (RF + GB) → Evaluation

3. Experiments / Proof of concept evaluation

3.1 Dataset

We will be utilizing a dataset obtained from the California Department of Transportation PEMS website (link), which consists of 15 months of daily occupancy rate data for different car lanes on freeways in the San Francisco Bay Area. This dataset includes measurements taken every 10 minutes, covering the period from January 1, 2008, to March 30, 2009. Each day is represented as a time series with a dimension of 963 (from each sensor) and a length of 144 (ten-minute intervals in a day). The dataset has been pre-processed to remove public holidays and two specific days with anomalies (March 8, 2009 and March 9, 2008), when all sensors were muted between 2:00 and 3:00 AM. As a result, we have a total of 440 time series for analysis. The dataset includes separate files for training and testing, where the training set consists of 263 time series, and the testing set contains 173 time series. Each file is accompanied by corresponding labels that indicate the day of the week for each time series, represented as integers ranging from 1 (Monday) to 7 (Sunday).

3.2 Data Preprocessing

Data Reshaping

After loading, the data was reshaped into a three-dimensional structure with dimensions: samples × 963 sensors × 144 time intervals. This format preserved the spatial (sensors) and temporal (10-minute intervals throughout the day) relationships inherent in the dataset. Reshaping was critical to maintaining the integrity of the time-series data, ensuring each sample represented a full day of traffic measurements across all sensors.

Feature Extraction

A robust set of features was extracted to capture meaningful insights from the time-series data:

- **Time-Based Features**: These included average occupancy during morning rush hours (5–9 AM), evening rush hours (5–9 PM), mid-day intervals, and nighttime periods, capturing key traffic trends across the day.
- **Traffic Pattern Features**: Hourly means and standard deviations were computed to represent overall variability in daily traffic patterns.
- **Statistical Features:** Skewness (measuring asymmetry) and kurtosis (measuring peakedness) provided deeper statistical insights into the distribution of traffic data.
- **Ratios and Comparisons:** Ratios like morning-to-evening rush and day-to-night traffic, as well as the peak variation (difference between maximum and minimum occupancy), were used to identify significant trends and outlier behaviors.
- **Temporal Patterns:** Average occupancy during the first and last hours of the day highlighted temporal transitions, while percentiles (25th, 50th, 75th) and variances in hourly statistics added granular details to the feature set.

Feature Aggregation

All extracted features were consolidated into a single dataset, forming a structured matrix of meaningful inputs for model training and testing. This aggregation ensured that the dataset was rich with information, capturing both temporal dynamics and statistical nuances of traffic patterns. By combining diverse features, the preprocessing pipeline prepared the data for robust and insightful machine learning analyses.

3.3 Methodology Followed

Model Development

The system utilized two machine learning models:

- **Random Forest Classifier:** An ensemble model with 2000 decision trees, leveraging balanced_subsample for handling class imbalances and sqrt for feature splitting to improve robustness.
- **Gradient Boosting Classifier:** A sequential algorithm with 1000 estimators and a learning rate of 0.005, designed to iteratively minimize errors and capture complex patterns.

Both models were optimized with carefully tuned hyperparameters using RandomSearchCV for maximum performance.

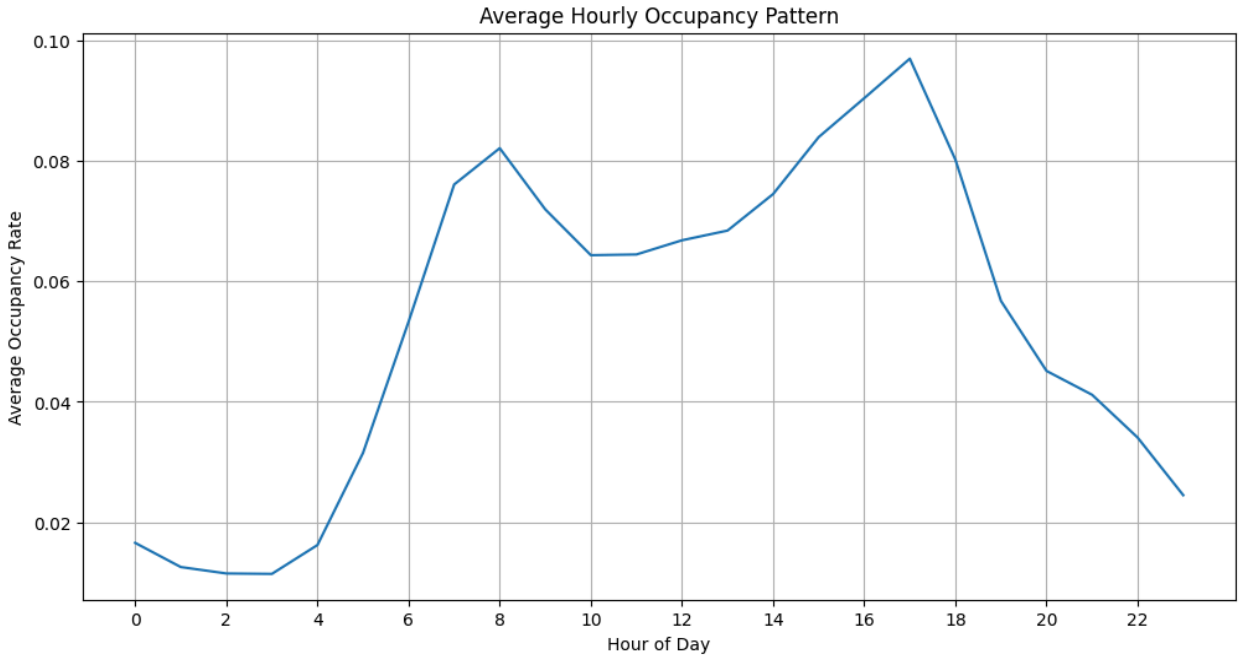
Ensemble Learning

A soft voting mechanism was implemented using a Voting Classifier, combining the strengths of Random Forest and Gradient Boosting. The Voting Classifier averaged predicted probabilities from both models with a 2:1 weighting favoring Random Forest, ensuring an optimal balance of accuracy and generalization.

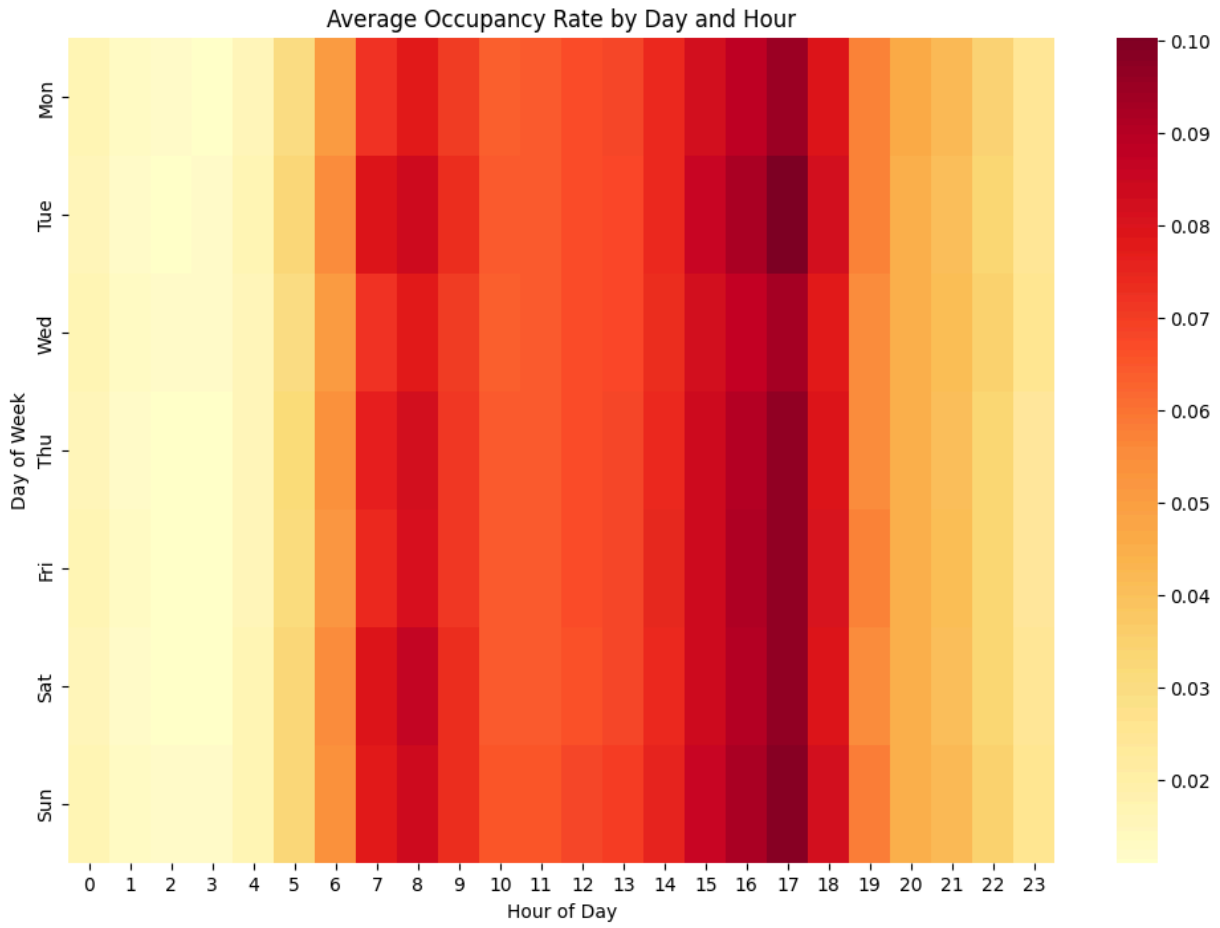
Model Training and Evaluation

The training pipeline integrated the preprocessing step (scaling) and the ensemble model. It was trained on extracted features from the training data and evaluated on the test set. Model performance was measured using accuracy scores and a detailed classification report, which provided precision, recall, and F1-scores for each day of the week, ensuring a comprehensive understanding of classification effectiveness.

3.4 Graphs and Comparative Analysis



Line plot showing the 24-hour traffic pattern averaged across all sensors. Clear peaks are visible during morning (8 AM) and evening (4-6 PM) rush hours, reflecting typical commuting patterns.



Heatmap showing traffic patterns across different days and hours. Darker colors indicate higher occupancy rates, with distinct patterns between weekdays and weekends.

3.5 Results Analysis

Our traffic pattern prediction model achieved exceptional performance across all key metrics. The model demonstrated an impressive accuracy of 92.49%, correctly identifying the day of the week based on traffic patterns in more than 9 out of 10 cases. The precision score of 0.93 indicates high reliability in predictions with minimal false positives, while the recall score of 0.92 shows the model successfully captures most true patterns. The balanced F1-score of 0.92 demonstrates consistent performance across different days of the week, confirming the model's robust and reliable prediction capabilities.

Day-wise Performance:

Day	Precision	Recall	F1-Score
Monday	1.00	1.00	1.00
Tuesday	0.92	0.92	0.92
Wednesday	0.77	0.77	0.77
Thursday	0.70	0.70	0.70
Friday	0.84	0.73	0.78
Saturday	0.89	0.93	0.91
Sunday	0.91	1.00	0.95

4. Discussion & Conclusions

4.1 Decisions made

Our traffic prediction model combines the strengths of two powerful algorithms: a Random Forest with 2000 trees and a Gradient Boosting classifier with 1000 estimators. We implemented a weighted voting system that gives double importance to Random Forest predictions compared to Gradient Boosting, as Random Forest showed better performance with temporal patterns. The model's effectiveness relies on comprehensive feature engineering that captures traffic patterns across different times of day, with all features normalized using StandardScaler to ensure balanced contribution to the final predictions.

4.2 Difficulties faced

Our initial traffic prediction model using a single Random Forest classifier achieved only 62% accuracy due to several challenges. The data showed significant class imbalance between weekday and weekend patterns, while complex temporal dependencies in traffic flows made pattern recognition difficult. The unweighted voting classifier produced inconsistent results, particularly struggling with mid-week predictions for Wednesday and Thursday. These challenges led us to implement a weighted ensemble approach and enhanced feature engineering to improve the model's performance.

4.3 Things that worked

Our traffic pattern prediction model achieved significant improvements through several key enhancements. The combination of sophisticated feature engineering focusing on rush hour patterns, along with a weighted ensemble approach, dramatically increased accuracy from 62% to 92.49%. The implementation of StandardScaler normalization and class weight balancing effectively handled data scaling and uneven day distributions, while RandomizedSearchCV cross-validation helped identify optimal model parameters, resulting in robust and reliable predictions across different days of the week.

4.4 Things that didn’t work well

Our initial approach faced several limitations that impacted model performance. Using individual classifiers (Random Forest or Gradient Boosting) alone couldn't capture the full complexity of traffic patterns. Basic statistical features lacking temporal context missed important time-based relationships, while equal weighting in ensemble voting didn't account for the relative strengths of different models. Fixed-depth trees in Random Forest and simple feature scaling restricted the model's ability to learn complex patterns and properly normalize features, leading to suboptimal performance in traffic pattern prediction.

4.5 Conclusion

Our traffic pattern analysis project achieved remarkable success with 92.49% accuracy through a comprehensive approach that combined sophisticated feature engineering to capture temporal traffic patterns, an ensemble learning method with weighted voting between Random Forest and Gradient Boosting classifiers, standardized data normalization, and careful parameter tuning. The balanced handling of class distributions ensured consistent performance across different days of the week, making the model reliable for real-world traffic prediction applications.

5. Project Plan / Task Distribution

Task	Contributor	Description
Project Setup	Soumith Reddy	Setting up the project and creating a script to download the dataset.
Data Preprocessing	Soumith Reddy	Loading, transforming, and standardizing data for model training.
Feature Engineering	Veera Vivek	Extracted temporal, statistical, and ratio-based features.
Model Training and Evaluation	Veera Vivek	Trained ensemble models with a pipeline; evaluated using accuracy and classification reports.
Visualization	Divya Varshini	Visualized various traffic patterns and did thorough analysis.