Report

Table of Contents

Table of Contents

Purpose

Project A – Predicting High Fire Risk Areas in Montreal

Objective: The objective of this project is to predict high fire risk areas by month in the city of Montreal, based on historical firefighter intervention data and additional open datasets.

Two main approaches

"Panel" Approach

Structure: Each building has one row per month (e.g., Jan 2018, Feb 2018, …, Dec 2022).

```
ID_UEV | YEAR | MONTH | Features... | fire_occurred
-----------------------------------------------------
001    | 2020 |   1   | ...         | 0
001    | 2020 |   2   | ...         | 0
001    | 2020 |   3   | ...         | 1
...
fire_occurred = 1 for the month of fire; 0 otherwise.
```

No fire_month column needed — you're modeling fire per building-month.

Good for temporal modeling, like survival analysis or monthly fire risk.

Fire Month or 13 Per-Building Row Approach

Structure: One row per building; fire_month = 1–12 if fire occurred, else 13.

```
ID_UEV | Features... | fire_month
---------------------------------
001    | ...         | 3
002    | ...         | 13
003    | ...         | 7
...
```

Simple classification task: which month is the fire, or no fire (13).

Only one row per building — no temporal unfolding.

fire_month = 13 explicitly encodes "no fire".

Data Cleaning and Merging Pipeline

evaluation_fonciere

Description

This script processes the Montréal property evaluation dataset, cleans it, and applies basic feature engineering to prepare it for downstream fire risk modeling.

How to Run

```
python ./dataprep/evaluation_fonciere.py
```

Output

- `datasets/cleaned/eval_cleaned.csv`
- `datasets/cleaned/eval_cleaned_feat_eng.csv` ← *(Recommended for modeling)*

Summary of Code Workflow

This script performs: - Cleaning of raw columns - Imputation of missing values - Creation of new features - Normalization of numeric variables

---

Main Steps

1. **Load raw dataset**
2. **Clean columns:**
   - `ANNEE_CONSTRUCTION`, `NOMBRE_LOGEMENT`, `ETAGE_HORS_SOL`, `SUPERFICIE_BATIMENT`
3. **Impute missing values using medians** at borough + building type level

4. **Feature engineering:**
    - AGE_BATIMENT
    - RATIO_SURFACE
    - DENSITE_LOGEMENT
    - HAS_MULTIPLE_LOGEMENTS
    - FIRE_FREQUENCY_ZONE
5. **Normalize** continuous features using MinMaxScaler
6. **Save output** to eval_cleaned_feat_eng.csv

Data Cleaning Summary

Column	Cleaning Strategy
ANNEE_CONSTRUCTION	Drop outliers (<1800 or >2025), impute with borough median (fallback: "unknown")
NOMBRE_LOGEMENT	Impute with (borough, building-type) median, fallback to borough median
ETAGE_HORS_SOL	Same imputation logic as above
SUPERFICIE_BATIMENT	Same imputation logic as above

Feature Engineering Summary

Feature Name	Description
AGE_BATIMENT	2025 - ANNEE_CONSTRUCTION
RATIO_SURFACE	SUPERFICIE_BATIMENT / SUPERFICIE_TERRAIN
DENSITE_LOGEMENT	NOMBRE_LOGEMENT / SUPERFICIE_BATIMENT
HAS_MULTIPLE_LOGEMENTS	1 if more than 1 housing unit, 0 otherwise
FIRE_FREQUENCY_ZONE	Number of buildings in the same NO_ARROND_ILE_CUM as proxy for risk

All continuous variables are normalized using MinMaxScaler.

Dropped/Excluded Columns

- _ANNEE_CONSTRUCTION_NUM
- Temporary columns used for imputation
- Potential future features like IS_UNKNOWN_YEAR, etc.

Final Output Columns

```
[
  'ID_UEV', 'CIVIQUE_DEBUT', 'CIVIQUE_FIN', 'NOM_RUE', 'SUITE_DEBUT',
  'MUNICIPALITE', 'ETAGE_HORS_SOL', 'NOMBRE_LOGEMENT', 'ANNEE_CONSTRUCTION',
  'CODE_UTILISATION', 'LETTRE_DEBUT', 'LETTRE_FIN', 'LIBELLE_UTILISATION',
  'CATEGORIE_UEF', 'MATRICULE83', 'SUPERFICIE_TERRAIN', 'SUPERFICIE_BATIMENT',
  'NO_ARROND_ILE_CUM', 'AGE_BATIMENT', 'RATIO_SURFACE', 'DENSITE_LOGEMENT',
  'HAS_MULTIPLE_LOGEMENTS', 'FIRE_FREQUENCY_ZONE'
]
```

Description: main_evaluation_feat_eng.py

Detailed Summary of `main_evaluation_feat_eng.py`

This script processes building and fire incident data to produce a geospatially enriched dataset for **fire risk modeling**. It includes spatial joins, temporal feature extraction, and fire frequency calculations at the zone level.

Goal of the Script Produce a feature-rich dataset at the building level that combines:

- Property evaluation data
- Fire incident history
- Geographic information (coordinates, zones)
- Temporal and structural features

Final output file: `evaluation_fire_coordinates_date_feat_eng_2.csv`

---

1. Load Datasets

- `eval_cleaned_feat_eng.csv` — Pre-cleaned building/property data
- `adresses.csv` — Street-level address coordinates
- `interventions_cleaned_with_has_fire.csv` — Fire incident reports

All file paths are dynamically resolved.

---

2. Preprocessing

Evaluation Data

- Cleans `CIVIQUE_DEBUT` (street number) and standardizes street names

- Copies original version for later merging

Address Data

- Combines `GENERIQUE` + `SPECIFIQUE` into `NOM_RUE_CLEAN`
- Converts address numbers for matching

Coordinate Assignment

- Merges buildings and addresses to assign `LATITUDE` and `LONGITUDE`
- Filters out rows without coordinates

Incident Data

- Filters rows where `DESCRIPTION_GROUPE` contains "INCENDIE"
- Converts date/time and builds GeoDataFrame
- Buffers incidents with a 100m radius to capture nearby buildings

---

3. Spatial Join

- Matches each building to nearby fire incidents using `gpd.sjoin()`
- Assigns:
  - `fire = True`
  - `fire_date`
  - `NOMBRE_UNITES` (fire truck count)
  - `CASERNE` (station name)

---

4. Merge Fire Info Back

- Merges fire records with **all buildings** using `ID_UEV`
- Fills nulls for non-fire rows
- Re-attaches coordinates from `addr_df`

---

5. Time Features

- Extracts:
  - `fire_month`, `fire_year`, `year_month`
  - `fire_season` (Winter, Spring, Summer, Fall)

---

6. Fire Zone Aggregates

- Computes per-zone fire counts for year 2024
- Computes:
  - `FIRE_COUNT_LAST_YEAR_ZONE`
  - `FIRE_RATE_ZONE` = fire count / buildings
- Applies `MinMaxScaler` to produce normalized versions

---

7. Missing Coordinates

- Flags rows missing latitude/longitude: `missing_coords`
- Compares fire rates between missing vs present coordinates

---

8. Feature Selection

Kept Features

- Structural: `AGE_BATIMENT`, `DENSITE_LOGEMENT`, `RATIO_SURFACE`
- Target: `fire`, `had_fire`, `fire_date`
- Time: `fire_month`, `fire_year`, `fire_season`, `year_month`
- Spatial: `NO_ARROND_ILE_CUM`, `LATITUDE`, `LONGITUDE`
- Fire Stats: `FIRE_COUNT_LAST_YEAR_ZONE`, `FIRE_RATE_ZONE`, and normalized versions

Dropped Features

- Redundant address fields: `CIVIQUE_DEBUT`, `ADDR_DE`, etc.
- Internal metadata: `MATRICULE83`, `CASERNE`, etc.
- Raw year field: `ANNEE_CONSTRUCTION` (keep `AGE_BATIMENT` instead)

---

9. Save Output

- Final cleaned file is saved to:

  `./datasets/cleaned/evaluation_fire_coordinates_date_feat_eng_2.csv`

Summary Stats

- ~664K total buildings
- ~295K matched to fire incidents
- ~202K rows missing coordinates (optional to drop)
- ~41 clean modeling-ready features

Fire Risk Monthly Panel Construction — Detailed Summary

This document explains the Python script that constructs a monthly panel dataset of buildings in Montréal enriched with fire event labels and engineered features. The final output is a panel of building-month combinations, used for predictive modeling of fire incidents.

Input & Output

Input:

evaluation_fire_coordinates_date_feat_eng_2.csv: Cleaned building dataset with fire incidents and coordinates

Output:

building_month_fire_panel_feat_eng.csv: Building-month panel with labels and features

Workflow Steps

Step 1: Setup Paths and Load Data

Define project root with Path.cwd().parents[1]

Load CSV using pandas

Preview data and confirm file existence

Step 2: Preprocessing & Cleaning

Convert fire_date to datetime

Create month feature

Drop records missing LONGITUDE, LATITUDE, or ID_UEV

Convert to GeoDataFrame with EPSG:32188 projection

Step 3: Construct Dense Panel

Extract unique buildings

Generate full range of monthly periods

Create full cartesian product → Dense panel (building × month)

Merge with building coordinates

Step 4: Label Fire Occurrence

Identify building-months with fire events

Create HAS_FIRE_THIS_MONTH binary column

Step 5: Time-Based Features

Extract month_num and year

Add lag features:

fire_last_1m, fire_last_2m, fire_last_3m

Add cumulative and rolling fire counts:

fire_cumcount, fire_rolling_3m, fire_rolling_6m, fire_rolling_12m

Add recency features:

has_fire_last_month, months_since_last_fire

Step 6: Merge Static Building Features

Extract one row per building from cleaned GeoDataFrame

Selected static columns:

MUNICIPALITE, ETAGE_HORS_SOL, NOMBRE_LOGEMENT, AGE_BATIMENT

CODE_UTILISATION, CATEGORIE_UEF, SUPERFICIE_TERRAIN, SUPERFICIE_BATIMENT

NO_ARROND_ILE_CUM, RATIO_SURFACE, DENSITE_LOGEMENT, HAS_MULTIPLE_LOGEMENTS

Fire zone features: FIRE_FREQUENCY_ZONE, FIRE_RATE_ZONE, FIRE_COUNT_LAST_YEAR_ZONE, etc.

Ensure alignment using valid building IDs

Step 7: Save Final Dataset

Export the resulting panel to CSV:

datasets/cleaned/building_month_fire_panel_feat_eng.csv

Next Steps

Add weather, crime, or inspection datasets for richer modeling

Evaluate model with recall and Precision@K

Consider borough-level aggregation or modeling using NO_ARROND_ILE_CUM

3. Model evaluation criteria

- Priority is given to recall over precision as we'd rather capture more fire risk including a few false negatives than miss high risk buildings
- Train set/Test set : we used a temporal split rather than random split Train = data before 2024 Test = data of 2024-2025

This is especially important because of the approximation of fire location due to data obfuscation to avoid leaking knowledge of fires into the train set.

4. Models tried

RandomForestClassifier

(Located in file EDA-incident-evaluation-fonciere.ipynb, for pipeline see instructions)

Target Variable

Y = P(Fire | X)

Where X includes:

- `log_terrain`
- `log_batiment`
- `log_etage_hors_sol`
- `log_numbre_de_logement`
- `ANNEE_CONSTRUCTION`
- `density`

Confusion Matrix

	Predicted False	Predicted True
Actual False	64,073	9,540
Actual True	8,273	50,871

Classification Report

Label	Precision	Recall	F1-score	Support
False	0.89	0.87	0.88	73,613
True	0.84	0.86	0.85	59,144
Accuracy			0.87	132,757
Macro avg	0.86	0.87	0.86	132,757
Weighted avg	0.87	0.87	0.87	132,757

ROC AUC: 0.936

LGBMClassifier

(Located in file Model-building.ipynb, for pipeline see instructions)

Target Variable

Y = P(Month of Fire | X)

Where X includes:

- ETAGE_HORS_SOL
- NOMBRE_LOGEMENT
- ANNEE_CONSTRUCTION
- SUPERFICIE_TERRAIN
- SUPERFICIE_BATIMENT
- LONGITUDE
- LATITUDE

Note: Properties with no recorded fire were assigned to month 13 to indicate the absence of fire incidents

Metric	Precision	Recall	F1-score	Support
Accuracy			0.676	132,757
Macro avg	0.457	0.348	0.388	132,757
Weighted avg	0.629	0.676	0.640	132,757

I then tried to train only on months 1-12:

Metric	Precision	Recall	F1-score	Support
Accuracy			0.434	58,954
Macro avg	0.439	0.433	0.434	58,954
Weighted avg	0.437	0.434	0.433	58,954

Monthly Fire Risk Prediction Using XGBoost

Script Location

`/dataprep/time_model_Xgboost.ipynb`

This script trains and evaluates a binary classifier to predict whether a fire will occur in a specific building in a given month. It utilizes a **dense panel dataset** with rich building-level and temporal-spatial features.

Data Pipeline

- **Input file:** `building_month_fire_panel_feat_eng.csv`
- **Granularity:** Monthly panel of all buildings
- **Target Variable:** `HAS_FIRE_THIS_MONTH` (0 or 1)

Feature Engineering

Structural & Geographic Features:

- MUNICIPALITE, ETAGE_HORS_SOL, NOMBRE_LOGEMENT, AGE_BATIMENT
- SUPERFICIE_TERRAIN, SUPERFICIE_BATIMENT, RATIO_SURFACE, DENSITE_LOGEMENT
- HAS_MULTIPLE_LOGEMENTS, CODE_UTILISATION, CATEGORIE_UEF
- NO_ARROND_ILE_CUM, BUILDING_COUNT

Zone-Level Fire Risk:

- FIRE_FREQUENCY_ZONE, FIRE_RATE_ZONE, FIRE_COUNT_LAST_YEAR_ZONE
- FIRE_RATE_ZONE_NORM, FIRE_COUNT_LAST_YEAR_ZONE_NORM

Temporal Lag Features:

- fire_last_1m, fire_last_2m, fire_last_3m
- fire_cumcount, fire_rolling_3m, fire_rolling_6m, fire_rolling_12m
- month_num, year

Model: XGBoostClassifier

- Handles class imbalance with scale_pos_weight
- Supports categorical variables with enable_categorical=True
- Optimized with:
  - n_estimators=200, max_depth=6, learning_rate=0.1
  - subsample=0.8, colsample_bytree=0.8

Evaluation (Default Threshold = 0.5)

Class	Precision	Recall	F1-score	Support
No Fire (0)	0.9903	0.7436	0.8494	3,674,405
Fire (1)	0.0243	0.4665	0.0461	50,239

- **Accuracy**: 73.99%
- **Macro F1**: 0.4477
- **Weighted F1**: 0.8379

Precision for fires is very low, but recall is moderate. Useful for prioritization, not alarms.

Threshold Optimization

Evaluated thresholds: `0.2` → `0.55` - Best **F2 Score** (recall-focused): **0.55**

Threshold	Precision	Recall	F2 Score
0.50	0.027	0.603	0.113
0.55	0.0262	0.378	**0.103**

Final Model Evaluation @ Threshold = 0.55

Class	Precision	Recall	F1-score	Support
No Fire	0.9896	0.8081	0.8897	3,674,405
Fire	0.0262	0.3780	0.0490	50,239

- **Confusion Matrix**:
  - True Negatives: ~2.97M
  - False Positives: ~700k
  - False Negatives: ~31k
  - True Positives: ~19k

  High false positives but acceptable for early warning.

Recommendations

- **Use threshold = 0.50–0.55** depending on recall vs precision preference
- **Precision is low**, so interpret predictions as risk levels
- **Use predicted probabilities** for prioritizing inspections
- Add weather, crime, or inspection features for improved separation

---

Binary vs Probabilistic Forecasting

Aspect	`model.predict()` (Binary)	`model.predict_proba()` + Threshold
Nature	Hard decision (0 or 1)	Probabilistic risk score (0.0–1.0)
Control over recall/precision	Fixed at 0.5	Fully tunable

Aspect	`model.predict()` (Binary)	`model.predict_proba()` + Threshold
Risk prioritization	Not possible	Buildings can be ranked
Early warning usability	Rigid	Flexible, interpretable
Best suited metric	Accuracy, F1	F2 Score, Precision@K, Recall

> **Recommendation:** Use predicted probabilities to forecast fire risk, prioritize inspections, and adapt to seasonal or strategic needs.

Next Steps

- Deploy as a **ranking tool** not a strict classifier
- Create dashboards that visualize monthly fire risk scores by building
- Use **precision@K** and **F2-score** as main evaluation metrics

---

Summary

XGBoost + panel-level fire features + lag history yields a decent early-warning fire detection model. Prioritize **recall and interpretability**, refine with more context over time. “““

5.

Forecasting and visualization

Monthly Fire Risk Forecasting with XGBoost – Project Summary

This script implements a full end-to-end pipeline for monthly fire risk prediction in Montréal using historical building data, engineered features, and an XGBoost classifier. It includes model training, threshold tuning, forecasting future risk, and generating interactive heatmaps for visual interpretation.

Data Overview

Input Files

- `building_month_fire_panel_feat_eng.csv`: Monthly panel of building features and historical fire labels.
- `evaluation_fire_coordinates_date_feat_eng_2.csv`: Coordinates and property attributes used for mapping.

Target Variable

- `HAS_FIRE_THIS_MONTH`: Binary variable indicating whether a fire occurred in a given month for a building.

Feature Engineering

Lag Features

- `fire_last_1m`, `fire_last_2m`, `fire_last_3m`: Fires in previous months per building.
- `fire_cumcount`, `fire_rolling_3m`, `fire_rolling_6m`, `fire_rolling_12m`: Cumulative or smoothed fire indicators.

Static and Zone Features

- Includes building characteristics (e.g., age, size, density), zone-level fire statistics, and encoded categories.

Model Training

Model: `XGBClassifier`

- Handles class imbalance using `scale_pos_weight`.
- Uses 200 estimators, max depth of 6, and categorical encoding.
- Trained on data from years 2023 and tested on 2024 data.

Evaluation and Threshold Tuning

Metrics

- Imbalanced classification: only ~1.3% of data are fire cases.
- Default threshold (0.5) leads to poor precision for fire class.
- High recall at low thresholds, but many false positives.

Optimal Threshold

- Evaluated using Precision, Recall, and F2 Score.
- Best **F2 Score 0.103** at **threshold = 0.55**, prioritizing recall over precision.
- Trade-off: higher recall (catching more fires) with acceptable false positives for early warning.

Future Panel Forecasting

- Generated fire risk predictions for the next **6 months** (panel built using unique buildings × future months).
- Static features frozen from latest available data.
- Missing time-dependent features (e.g., recent fires) set to zero as placeholders.

Visualization

Interactive Folium Heatmaps

- **Map Layers**: Fire risk scores overlaid as a heatmap with a red–orange–yellow gradient.
- **Monthly Maps**: Visuals generated per month using ipywidgets for interactive selection.
- **Thresholding**: Only predictions with risk 0.5 included in visual maps.
- **Legend**: Custom HTML element included for interpretation.

Model Deployment Steps

1. Save the trained XGBoost model using `joblib`.
2. Load future panel, run predictions using saved model.
3. Merge predictions with coordinates.
4. Save both `.csv` predictions and `.html` maps.

Recommendations

Use Threshold = 0.55

- Balanced fire detection rate (recall ~38%) with manageable false positives.
- Suitable for **early warning and resource prioritization** (not final decision making).

Improve Model By:

- Adding external data (e.g., weather, past interventions, crime data).
- Testing ensemble models.
- Ranking buildings by predicted probability for inspection priority.

Output Files

- `fire_risk_heatmap.html`: Interactive map for current/future fire risks.
- `future_fire_risk_panel_6m.csv`: Feature panel for 6-month forecast.
- `future_panel_predictions.csv`: Fire probabilities for each building-month.
- `xgb_fire_model.pkl`: Trained XGBoost model for reuse.
- `fire_risk_map_august_2025.html`: Example monthly forecast visualization.
