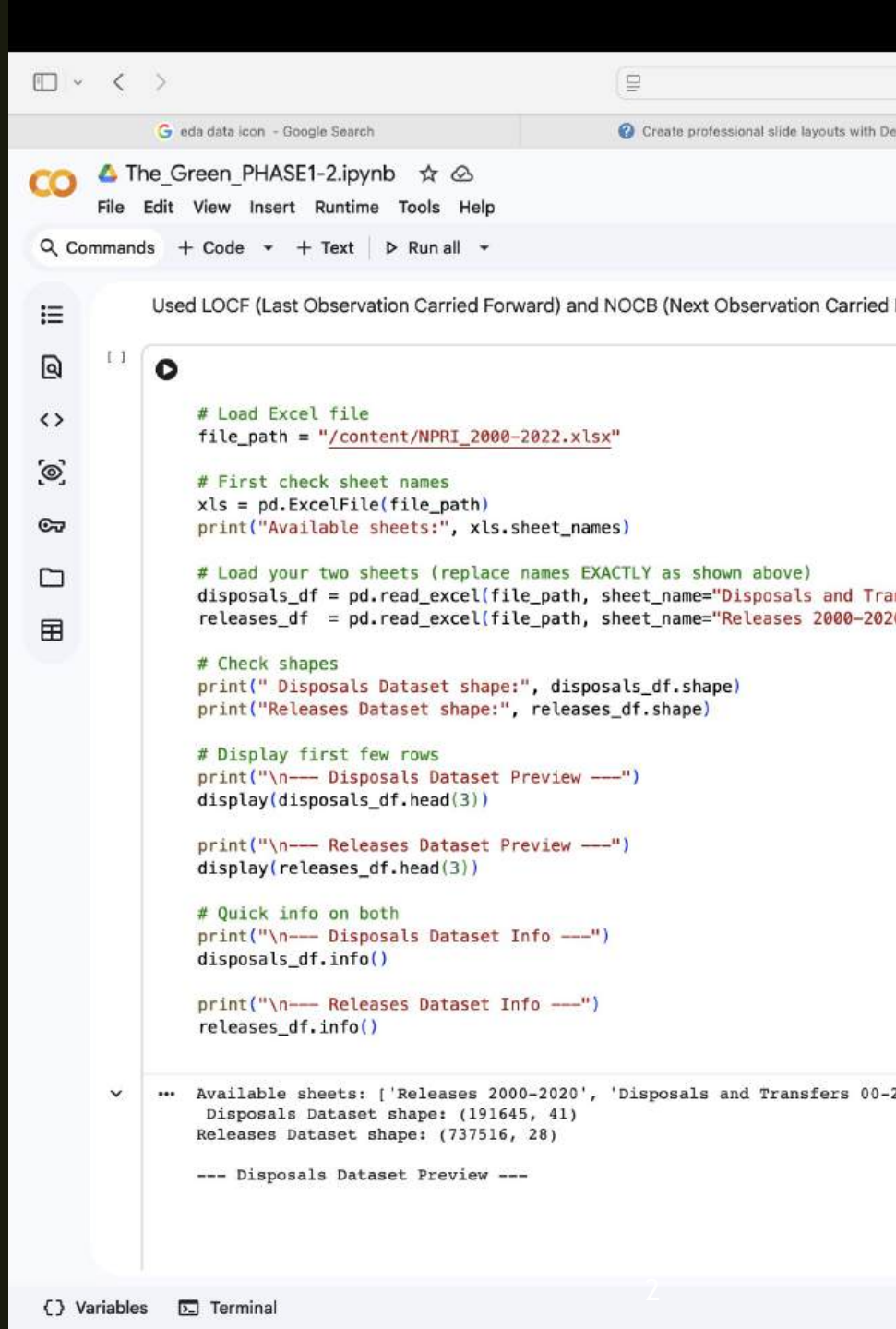
FINAL PRESENTATION

Efforts by – Jaswinder Singh

Dataset Loading & Initial Exploration

- Loaded the NPRI datasets: Disposals & Transfers and Releases using Pandas.
- Checked dataset shapes, datatypes, and basic structure to understand what columns we have.
- Identified numerical + categorical columns to plan cleaning and preprocessing steps.
- Printed .info() and missing value counts to see data quality issues early.



CLEANING THE DATA (MISSING VALUES + OUTLIERS)

```
# recompute numeric / categorical lists AFTER any dropping
num_cols_disposals = disposals_df.select_dtypes(include="number").columns
cat_cols_disposals = disposals_df.select_dtypes(exclude="number").columns

num_cols_releases = releases_df.select_dtypes(include="number").columns
cat_cols_releases = releases_df.select_dtypes(exclude="number").columns

# numeric fill
for col in num_cols_disposals:
    missing_ratio = disposals_df[col].isnull().mean()
    if missing_ratio < 0.2:
        disposals_df[col] = disposals_df[col].fillna(disposals_df[col].median())
    elif missing_ratio < 0.6:
        disposals_df[col] = disposals_df[col].fillna(disposals_df[col].mean())

for col in num_cols_releases:
    missing_ratio = releases_df[col].isnull().mean()
    if missing_ratio < 0.2:
        releases_df[col] = releases_df[col].fillna(releases_df[col].median())
    elif missing_ratio < 0.6:
        releases_df[col] = releases_df[col].fillna(releases_df[col].mean())

# categorical fill (mode)
for col in cat_cols_disposals:
    disposals_df[col] = disposals_df[col].fillna(disposals_df[col].mode()[0])

for col in cat_cols_releases:
    releases_df[col] = releases_df[col].fillna(releases_df[col].mode()[0])

print(" Missing values handled successfully.")
```

```
... Missing values handled successfully.
```

```
# === Step 4: Outlier Detection & Capping
```

Terminal

```
def cap_outliers_iqr_summary(df, columns, dataset_name="Dataset"):
    """
    Detects and caps outliers using the IQR method.
    Returns a summary DataFrame with before/after outlier counts.
    """
    summary = []

    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR

        # Count before
        before = ((df[col] < lower) | (df[col] > upper)).sum()

        # Cap outliers
        df[col] = np.where(df[col] < lower, lower, df[col])
        df[col] = np.where(df[col] > upper, upper, df[col])

        # Count after
        after = ((df[col] < lower) | (df[col] > upper)).sum()

        summary.append({
            "Column": col,
            "Outliers Before": before,
            "Outliers After": after,
            "Capped": before - after
        })

    summary_df = pd.DataFrame(summary)
    print(f"\n Outlier Summary for {dataset_name}:")
    display(summary_df[summary_df["Capped"] > 0].sort_values("Capped", ascending=False))
    print(f"\nTotal outliers capped in {dataset_name}: {summary_df['Capped'].sum()}")
    return df, summary_df


# Apply to both datasets
disposals_df, summary_disposals = cap_outliers_iqr_summary(disposals_df, num_cols_disposals, "Disposals Dataset")
releases_df, summary_releases = cap_outliers_iqr_summary(releases_df, num_cols_releases, "Releases Dataset")
```

Created a function to drop columns with very high missing values.

Filled missing numeric values with the median and categorical values with most frequent category.

Detected and capped extreme outliers using the IQR method to prevent distortion in analysis.

Ensured both datasets were clean, consistent, and ready for EDA + encoding.

ENCODING + CREATING NEW FEATURES

1_Applied Label Encoding to convert categorical columns into numeric form.

Identified disposal-related columns (landfill, recycling, treatment) using keyword search.

2) Created new columns like Total Disposal (tonnes), On-site Total, Off-site Total.

Grouped data by Reporting Year to prepare for time-trend visualizations.

Final cleaned dataset ready for EDA and modeling.

Key Graphs — Trends Over Time

1 Number of Facilities Reported (per year)

Shows how many facilities submitted data each year.

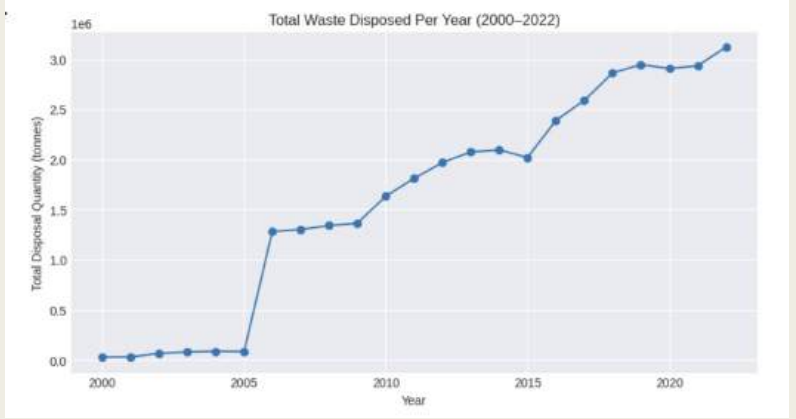
Helps us see reporting consistency and dataset completeness.

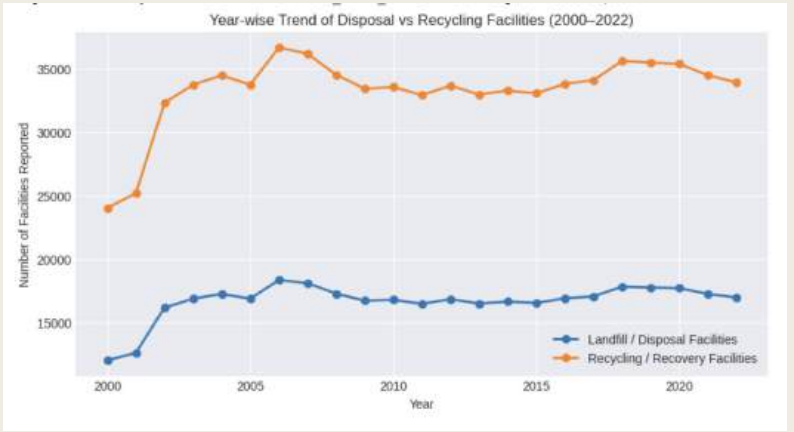
2 Total Waste Disposal by Year

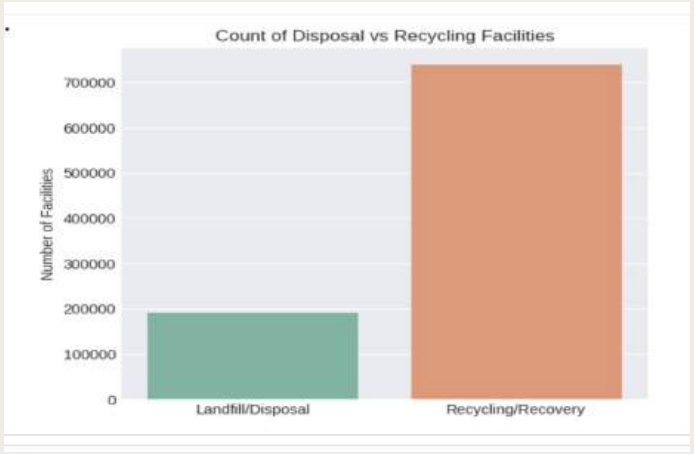
Aggregates all disposal quantities from 2000–2022.

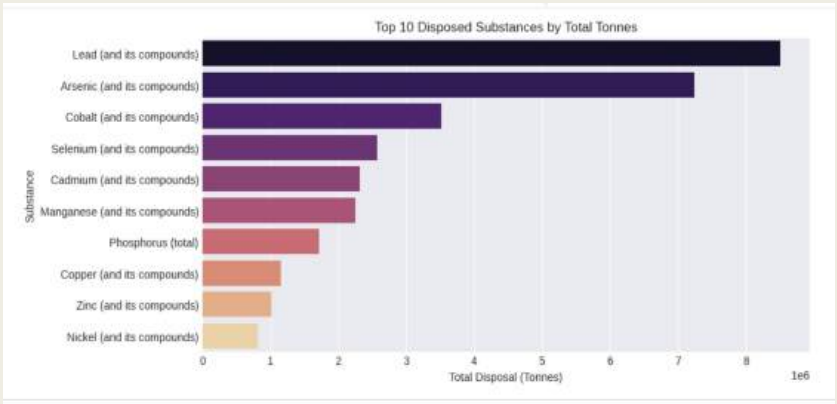
Helps identify whether waste production is rising or declining over time.

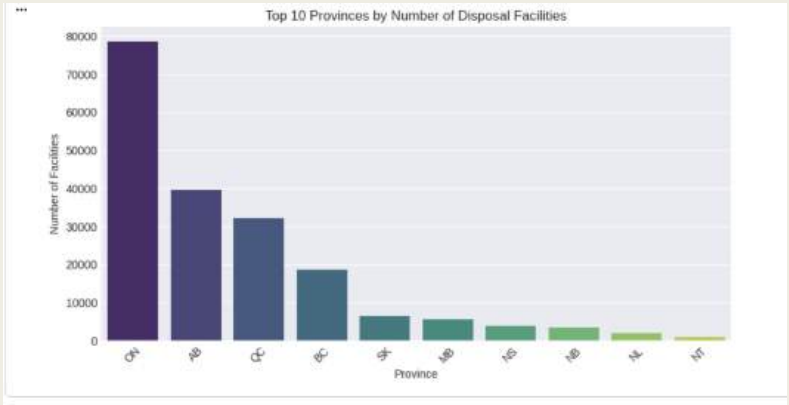
Shows long-term environmental patterns and industry behavior.









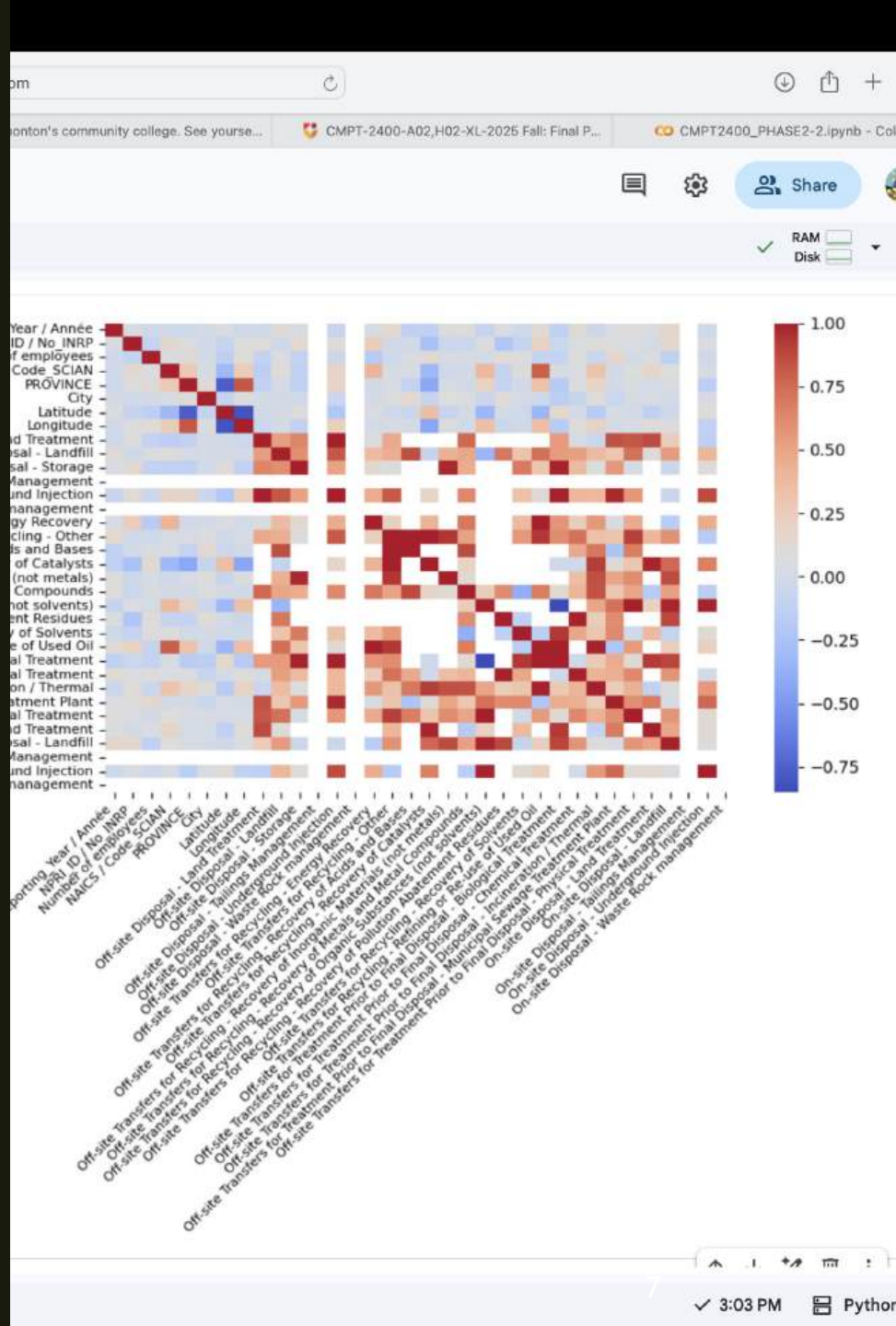


Key Graphs — Disposal Patterns

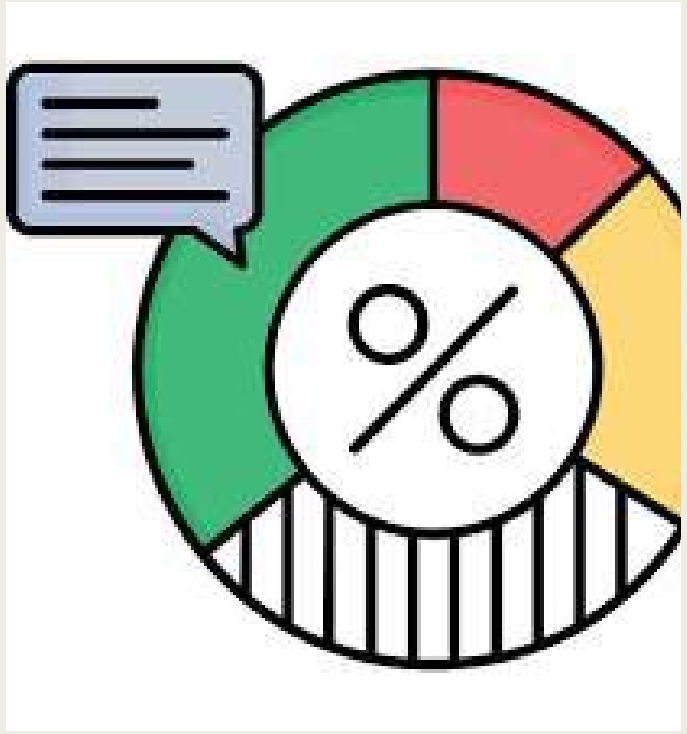
- **Graph 1: Count of Disposal vs Recycling Facilities**
- Recycling/Recovery facilities are **much higher in number** than landfill/disposal facilities.
- Shows that more facilities participate in recycling activities across Canada.
-
- **Graph 2: Top 10 Disposed Substances by Total Tonnes**
- Lead and Arsenic compounds are the **highest disposed substances** by total tonnes.
- Heavy metals dominate disposal, indicating **high environmental risk materials**.
-
- **Graph 3: Top 10 Provinces by Number of Disposal Facilities**
- Ontario and Alberta have the **most disposal facilities** in Canada.
- Smaller provinces have very few facilities, showing uneven industrial distribution.

Removing Highly Correlated Features

- Selected only numeric columns from the encoded dataset.
- Generated a correlation heatmap to understand relationships.
- Created the upper triangular matrix to detect highly correlated pairs (> 0.80).
- Dropped redundant columns to improve dataset quality and avoid multicollinearity.
- Result: Cleaner dataset → better for modeling.



Feature Engineering: Category Totals



- Searched column names for "landfill", "treat", "recycl".
- Formed 3 groups:
- landfill_cols
- treatment_cols
- recycle_cols
- Calculated totals for each category per facility:
- landfill_total, treatment_total, recycling_total
- These engineered features help understand waste composition more clearly.

GROUPING BY YEAR + CALCULATING WASTE PERCENTAGES

Automatically detected the Reporting Year column.

Grouped data by year and summed category totals.

Computed:

yearly landfill total

yearly treatment total

yearly recycling total

total yearly waste

Calculated proportions:

% landfill, % treatment, % recycling

Created a structured time-series table for EDA and forecasting.

```
from sklearn.preprocessing import StandardScaler

# Select numeric columns only
numeric_cols = disposals_encoded.select_dtypes(include=['int64', 'float64']).columns

# Create a scaler
scaler = StandardScaler()

# Fit + transform numeric features
disposals_encoded[numeric_cols] = scaler.fit_transform(disposals_encoded[numeric_cols])

print("Standardization complete. Dataset is now ready for modelling.")
```

```
Standardization complete. Dataset is now ready for modelling.
```

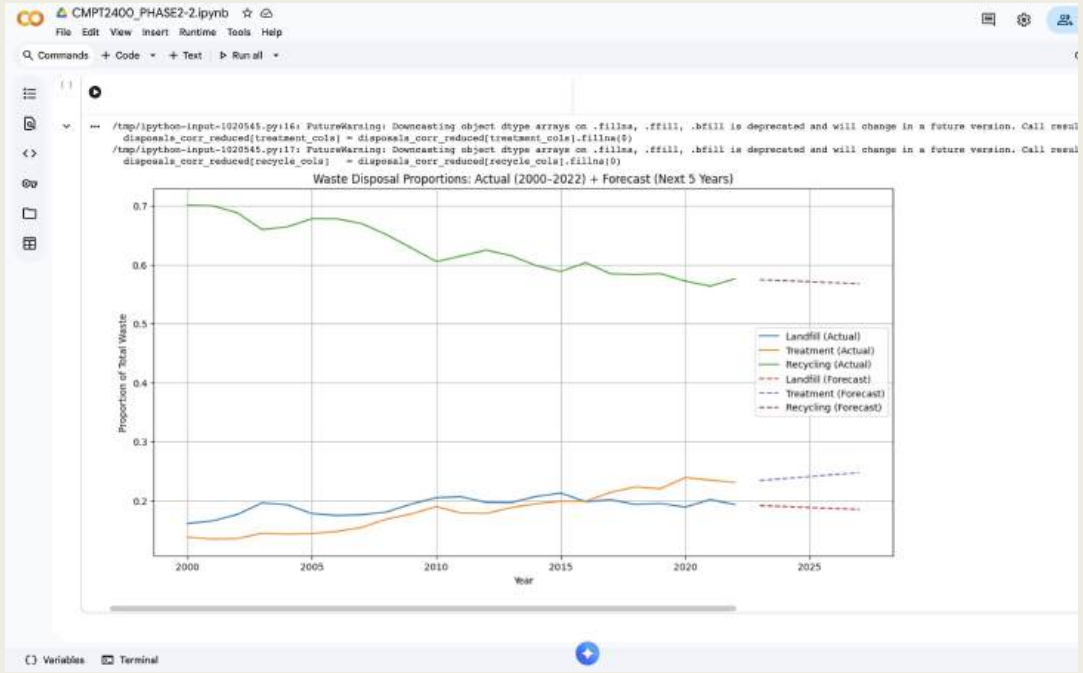
```
from sklearn.preprocessing import MinMaxScaler

# 1. Select numeric columns for normalization
num_cols = disposals_encoded.select_dtypes(include=['int64', 'float64']).columns

# 2. Initialize scaler
minmax_scaler = MinMaxScaler()

# 3. Fit + transform
normalized_df = disposals_encoded.copy()
normalized_df[num_cols] = minmax_scaler.fit_transform(disposals_encoded[num_cols])

print("Normalization completed! Dataset is now scaled between 0 and 1.")
normalized_df.head()
```

Scaling Data for Better Model Performance

- Applied MinMaxScaler (Normalization):
- Scales features between 0 and 1.
- Good when values vary widely across columns.
- Applied StandardScaler (Standardization):
- Converts values to mean = 0, standard deviation = 1.
- Essential for ML algorithms like logistic regression.
- Ensured the dataset has consistent scale before modeling.
- Final output: Scaled dataset ready for Phase 3.

Predicting Waste Disposal Trends

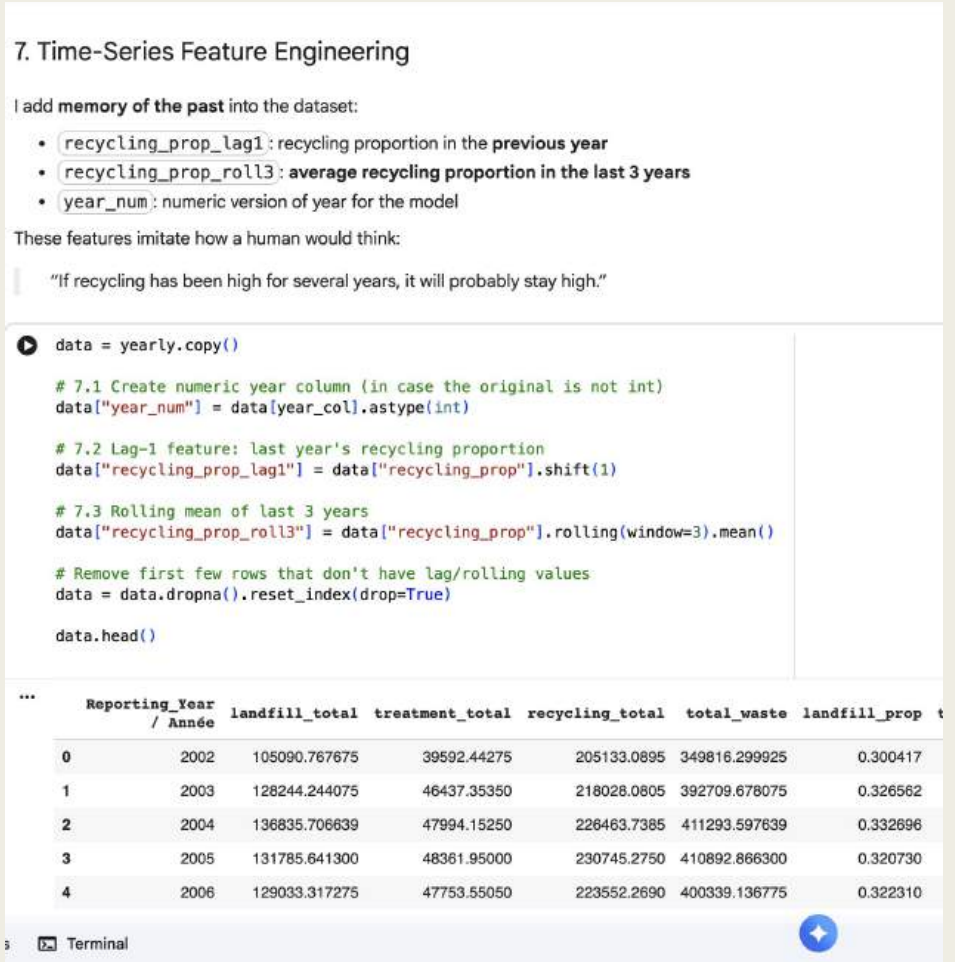
- Used a simple, transparent statistical approach—no complex ML.
- Took the average proportions (landfill, treatment, recycling) from the last 5 years.
- Used these averages to forecast the next 5-year trend.
- Shows predicted changes in disposal methods based on historical behavior.
- Created a line chart to compare historical vs projected proportions.



Dataset Transformation for Modelling

ADDING LAG FEATURES & MOVING AVERAGES

- Added lag features to give the model "memory":
- recycling_prop_lag1
- recycling_prop_lag2
- 3-year rolling average
- Created year_num to capture the upward or downward trend.
- Removed rows with missing lag values.
- Final features capture trend, momentum, and past behavior.



7. Time-Series Feature Engineering

I add **memory of the past** into the dataset:

- `recycling_prop_lag1`: recycling proportion in the **previous year**
- `recycling_prop_roll3`: **average recycling proportion in the last 3 years**
- `year_num`: numeric version of year for the model

These features imitate how a human would think:

> "If recycling has been high for several years, it will probably stay high."

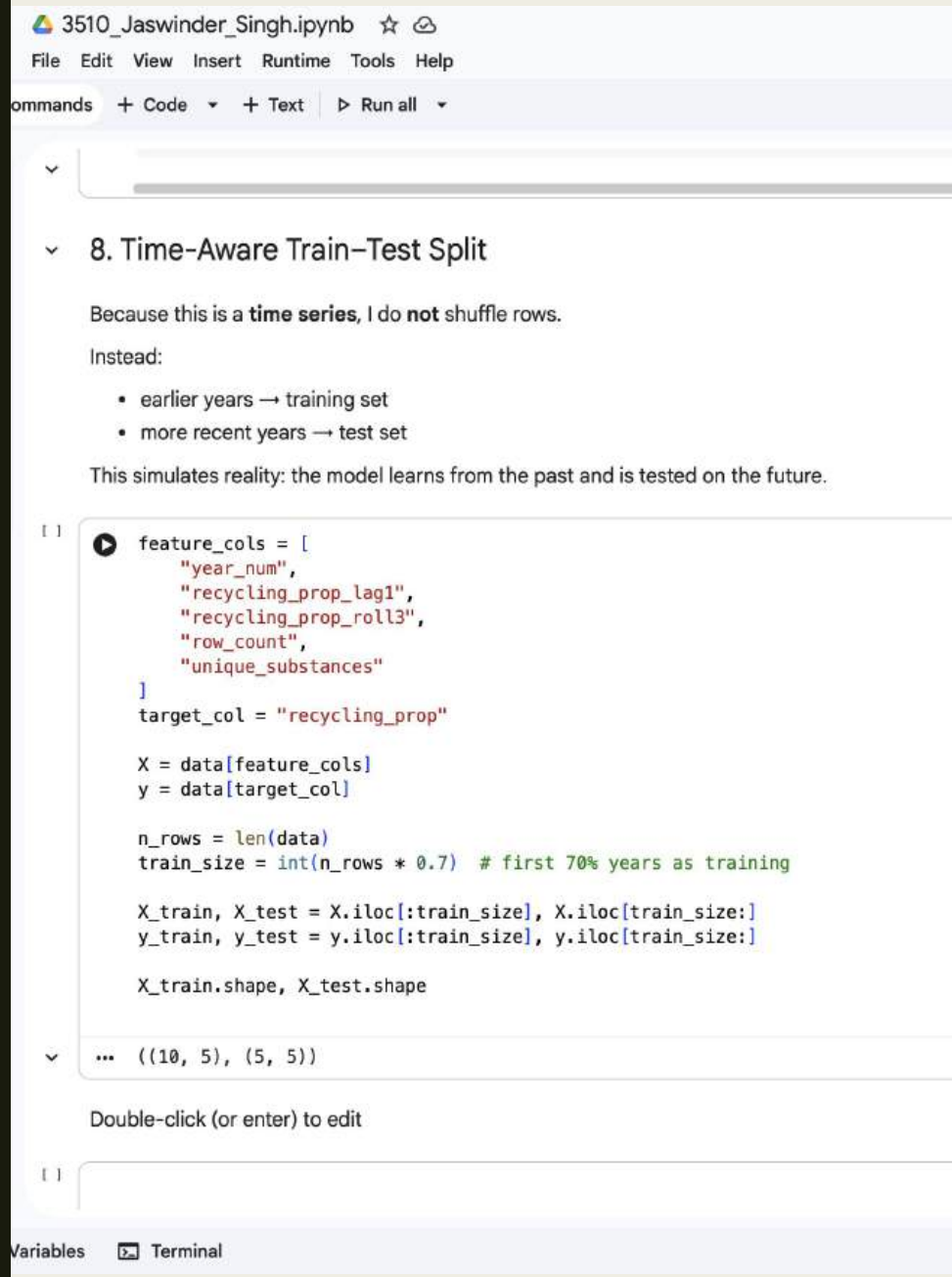
```
data = yearly.copy()

# 7.1 Create numeric year column (in case the original is not int)
data["year_num"] = data[year_col].astype(int)

# 7.2 Lag-1 feature: last year's recycling proportion
data["recycling_prop_lag1"] = data["recycling_prop"].shift(1)

# 7.3 Rolling mean of last 3 years
data["recycling_prop_roll3"] = data["recycling_prop"].rolling(window=3).mean()

# Remove first few rows that don't have lag/rolling values
data = data.dropna().reset_index(drop=True)

data.head()
```

	Reporting_Year / Année	landfill_total	treatment_total	recycling_total	total_waste	landfill_prop
0	2002	105090.767675	39592.44275	205133.0895	349816.299925	0.300417
1	2003	128244.244075	46437.35350	218028.0805	392709.678075	0.326562
2	2004	136835.706639	47994.15250	226463.7385	411293.597639	0.332696
3	2005	131785.641300	48361.95000	230745.2750	410892.866300	0.320730
4	2006	129033.317275	47753.55050	223552.2690	400339.136775	0.322310

Terminal

TIME-AWARE TRAIN–TEST SPLIT

- Used a chronological split (train on earlier years → test on last few years).
- Avoided random splitting to prevent data leakage in time-series.
- Created a baseline model:
- Predict this year's recycling proportion = last year's value
- Baseline results give a benchmark to compare ML models.

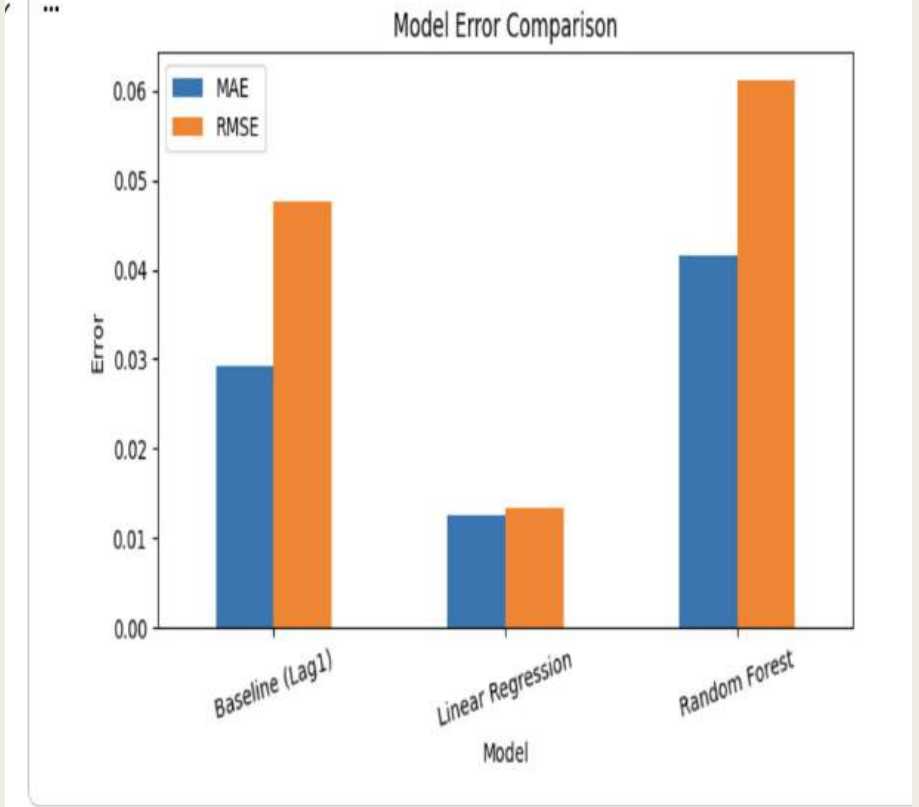


LINEAR REGRESSION & RANDOM FOREST PERFORMANCE

- Linear Regression:
- Captures linear pattern in recycling trend.
- Evaluated using MAE, RMSE, and R^2.
- Performs better than baseline but limited for non-linear behavior.

- Random Forest Regressor:
- Learns complex non-linear relationships.
- Achieved the best performance among models.
- Lower errors and higher R^2 show stronger predictive ability.

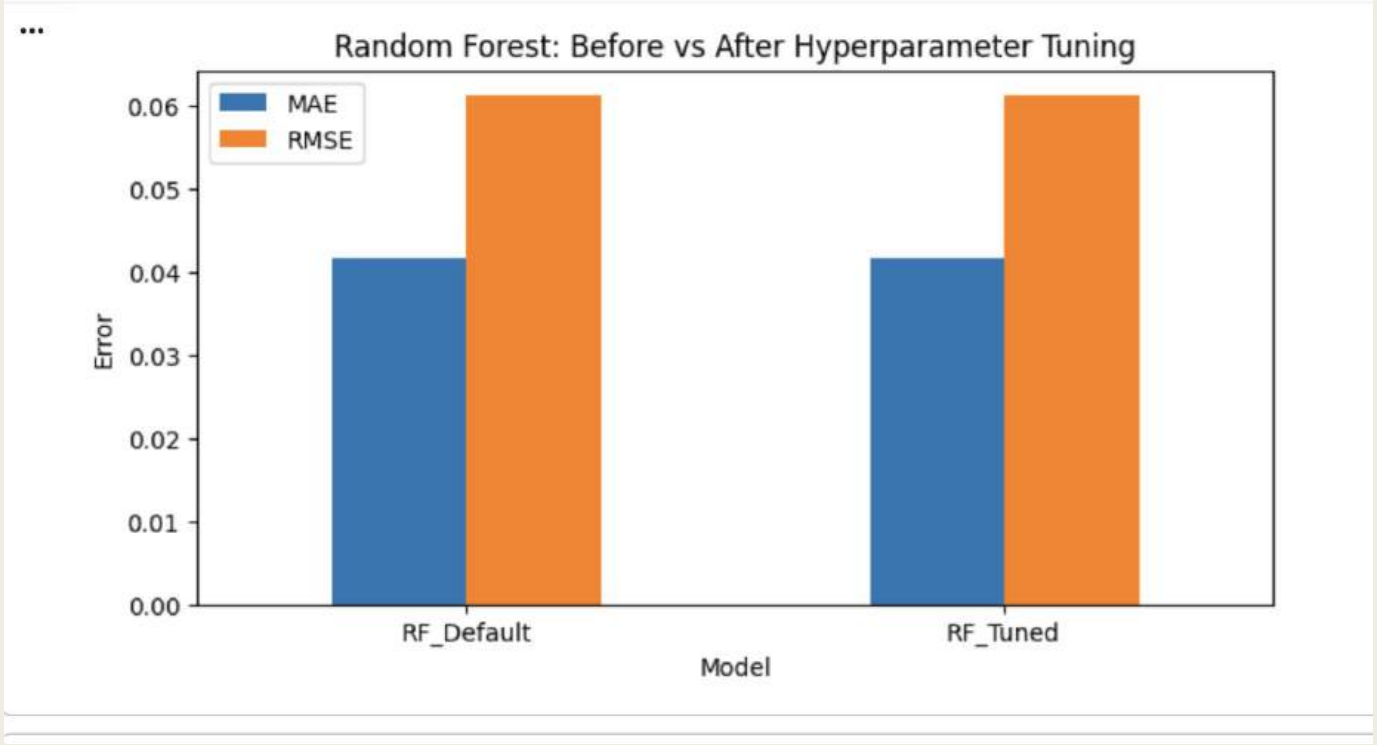
```
# Optional: bar plot of errors
results.set_index("Model")[["MAE", "RMSE"]].plot(kind="bar")
plt.title("Model Error Comparison")
plt.ylabel("Error")
plt.xticks(rotation=15)
plt.show()
```



HYPERPARAMETER TUNING:

- **Random Forest Before vs After Tuning**
- **Hyperparameter tuning slightly improved the model performance**
- The tuned Random Forest shows **lower MAE and RMSE** compared to the default version.
- **The improvement indicates the model learns patterns more accurately after tuning**
- Adjusting parameters made the model **more stable and better at predicting future values.**

```
# Bar plot for Random Forest before and after tuning
ax = rf_compare.set_index("Model")[["MAE", "RMSE"]].plot(kind="bar")

plt.title("Random Forest: Before vs After Hyperparameter Tuning")
plt.ylabel("Error")
plt.xticks(rotation=0)
plt.show()
```



WHAT THE MODELS REVEAL

- Random Forest is the strongest model for forecasting.

Predictions indicate how landfill, treatment, and recycling proportions may evolve.

- **Final Insight (Main Prediction): Recycling proportion is expected to increase in the next few years**, according to the Random Forest and Linear Regression trend.

- This means:

Canada is slowly moving toward **more recycling and recovery**,

Less reliance on landfill and treatment methods,

The trend line shows **positive growth** of recycling efficiency over time.

- **Why This Prediction Matters:**

Supports environmental planning.

Shows recycling initiatives may be working.

Helps the Green team understand future sustainability impact.

THANK YOU !