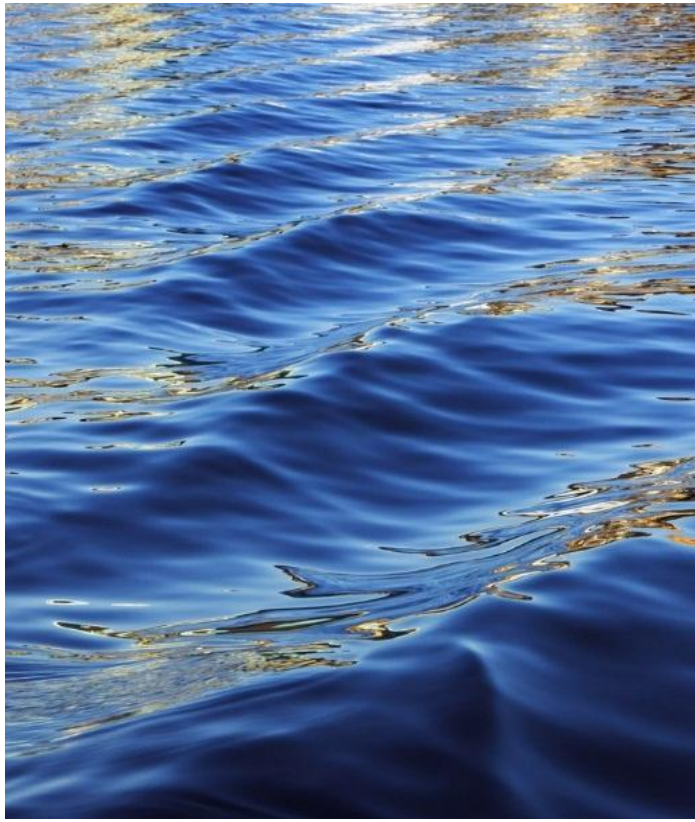
Climate Change in Canada: Historical Analysis and Future Projections

Aysegul Dahi



Index

The Goal of the Project

The goal of this project is to predict key climate indicators for Canada and the world using advanced machine learning techniques. I used historical data from 1968 to 2024 (except for CO_2 emissions data, which covers a different range) to train models and generate predictions for the future period of 2025 to 2030.

The project focuses on forecasting:

- Sea ice loss (for Canada),
- Temperature change (for Canada),
- Precipitation trends (for Canada),
- and CO_2 emissions (globally).

I merged multiple datasets from different sources to create a comprehensive view of historical climate patterns. For modeling, applied and compared three different machine learning models: Prophet, Gradient Boosting, and Random Forest. Then I visualized the historical datasets and predictions with interactive maps and various of charts in Streamlit app.

While sea ice, temperature, and precipitation predictions are centered on Canada, CO_2 emissions predictions are global — allowing us to visualize Canada's position in the worldwide climate challenge.

This project, aims to support policymakers, researchers, and the public by providing accessible predictions that highlight future climate risks and opportunities for proactive environmental action.

Datasets used in this Project and Their Resources

File	Name	Resource	Link
climate_history.csv	Eighty years of Canadian climate data	Kaggle	https://www.kaggle.com/datasets/aturner374/eighty-years-of-canadian-climate-data
co2-emissions-and-gdp.csv	Change in CO_2 emissions and GDP	Our World in Data	https://ourworldindata.org/grapher/co2-emissions-and-gdp
co-emissions-per-capita.csv	Per capita CO_2 emissions	Our World in Data	https://ourworldindata.org/grapher/co-emissions-per-capita
sea_ice_loss.csv	Sea ice in Canada	Environment and Climate Change Canada (ECCC)	https://www.canada.ca/en/environment-climate-change/services/environmental-indicators/sea-ice.html
September-trends-Arctic-en.csv	Sub-region September sea ice area decadal trends, Canadian Arctic domain, 1968 to 2024	Environment and Climate Change Canada (ECCC)	https://www.canada.ca/en/environment-climate-change/services/environmental-indicators/sea-ice.html
summer-trends-NCW.csv	Sub-region summer sea ice area trends, Northern Canadian Waters, 1968 to 2024	Environment and Climate Change Canada (ECCC)	https://www.canada.ca/en/environment-climate-change/services/environmental-indicators/sea-ice.html
Temperature-change-annual-en.csv	Temperature change in Canada	Environment and Climate Change Canada (ECCC)	https://www.canada.ca/en/environment-climate-change/services/environmental-indicators/temperature-change.html

Data Cleaning and Merging

Cleaning Sea Ice Loss Datasets
•Loaded historical *September* and *Summer* sea ice area datasets data_cleaning.
•Standardized column names to include the season (e.g., Foxe Basin (Summer), Foxe Basin (September)).
•Converted columns to numeric types to handle missing or corrupted values.
•Merged September and Summer trends into one dataset based on **Year**.

Cleaning Additional Climate Datasets
•Loaded *Sea Ice Loss*, *Temperature Change*, and *CO_2 Emissions per Capita* datasets merge_all_datasets.
•Renamed inconsistent columns for clarity and consistency.
•Ensured all "Year" columns are integer type across datasets.

Handling Missing Data
•Checked for missing values.
•Filled missing temperature, CO_2 emissions, and sea ice values using the **median** of each column.
•Applied **forward fill** for time-series consistency.

Feature Engineering
•Created a new feature: **Temperature Change in Fahrenheit** for easier interpretation.

Final Merging
•Merged all cleaned datasets into a **single master dataset** from **1968 to 2024**.
•Saved the final cleaned dataset as: csv_files/cleaned_datasets/final_climate_dataset.csv

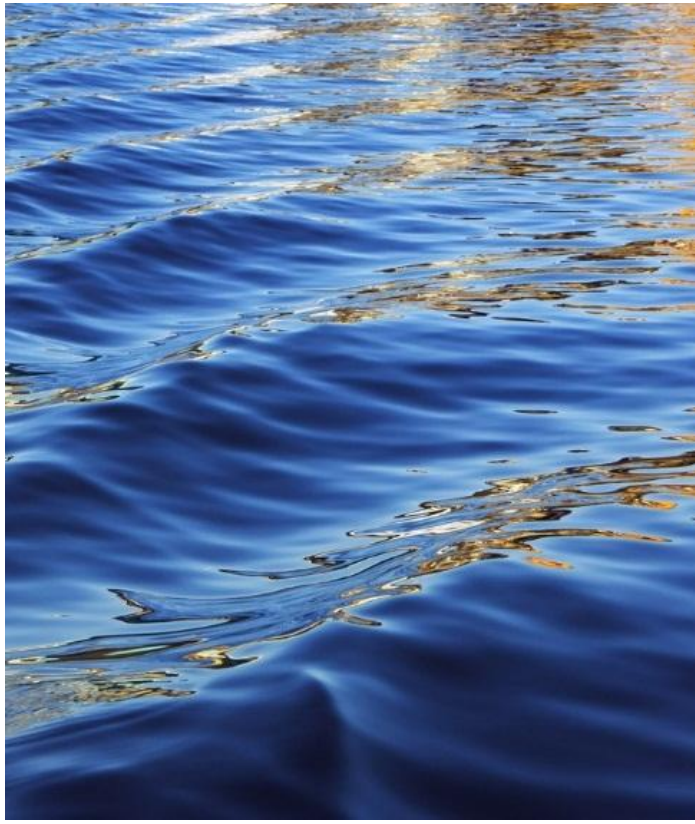
Data Cleaning and Merging

Screenshot of the Codes

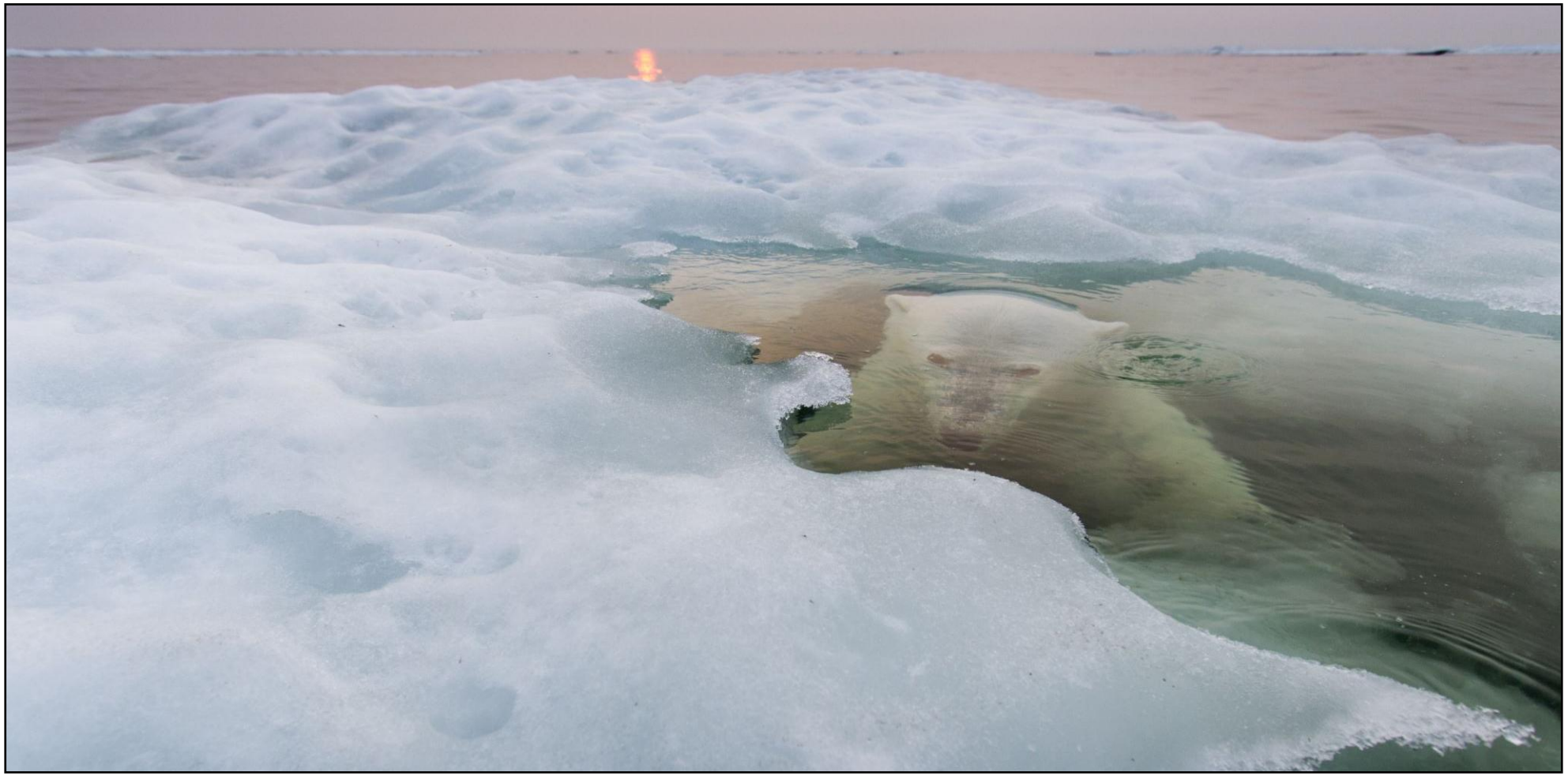
```python
import pandas as pd

# Load Datasets
september_trends_path = "csv_files/raw_datasets/September-trends-Arctic-en.csv"
summer_trends_path = "csv_files/raw_datasets/summer-trends-NCW.csv"

september_trends = pd.read_csv(september_trends_path)
summer_trends = pd.read_csv(summer_trends_path)

print("September Trends Shape:", september_trends.shape)
print("Summer Trends Shape:", summer_trends.shape)

#  Rename Columns (Fixing for Summer and September)
september_trends = september_trends.rename(columns={
    "Foxe Basin": "Foxe Basin (September)",
    "Kane Basin": "Kane Basin (September)",
    "Baffin Bay": "Baffin Bay (September)",
    "Beaufort Sea": "Beaufort Sea (September)",
    "Canadian Arctic Archipelago": "Arctic Archipelago (September)",
    "Canadian Arctic domain (thousands of square kilometres)": "Arctic Domain (September)"
})

summer_trends = summer_trends.rename(columns={
    "Foxe Basin": "Foxe Basin (Summer)",
    "Kane Basin (thousands of square kilometres)": "Kane Basin (Summer)",
    "Baffin Bay (thousands of square kilometres)": "Baffin Bay (Summer)",
    "Beaufort Sea (thousands of square kilometres)": "Beaufort Sea (Summer)",
    "Canadian Arctic Archipelago": "Arctic Archipelago (Summer)",
    "Hudson Bay (thousands of square kilometres)": "Hudson Bay (Summer)",
    "Hudson Strait (thousands of square kilometres)": "Hudson Strait (Summer)",
    "Davis Strait (thousands of square kilometres)": "Davis Strait (Summer)",
    "Northern Labrador Sea (thousands of square kilometres)": "Northern Labrador Sea (Summer)"
})
```

```python
# Check missing values count
missing_values = merged_full.isnull().sum()
print("Missing Values in Each Column:\n")
print(missing_values[missing_values > 0])

# Fill missing values using median for numerical columns
merged_full.fillna({
    "Temperature Change (°C)": merged_full["Temperature Change (°C)"].median(),
    "CO2 Emissions (per capita)": merged_full["CO2 Emissions (per capita)"].median(),
    "Total Sea Ice Area (millions km²)": merged_full["Total Sea Ice Area (millions km²)"].median()
}, inplace=True)

# Forward-fill for time-series data
merged_full.ffill(inplace=True)

# Convert Temperature Change to Fahrenheit & Keep Both
merged_full["Temperature Change (°F)"] = merged_full["Temperature Change (°C)"] * 9/5 + 32

output_dir = "csv_files"
os.makedirs(output_dir, exist_ok=True)
output_path = os.path.join(output_dir, "cleaned_datasets/final_climate_dataset.csv")

merged_full.to_csv(output_path, index=False)

print(merged_full.info())
print(merged_full.head())

print("Cleaned dataset saved successfully at: {output_path}")
```

Machine Learning Models and Scores



Predicting the Sea Ice



Predicting the Sea Ice

Modeling Sea Ice Loss (Summer & September)

Historical Years Used:
1968 to 2024

Prediction Years:
2025 to 2030

Features Used:
•Only **Year** was used as a feature.
•To capture non-linear trends, **polynomially expanded** Year into second-degree features (Year, Year²).

Target Variables (Regions Predicted):
•Summer Trends:
Foxe Basin (Summer), Kane Basin (Summer), Baffin Bay (Summer), Beaufort Sea (Summer),
Arctic Archipelago (Summer), Hudson Bay (Summer), Hudson Strait (Summer), Davis Strait (Summer),
• Northern Labrador Sea (Summer)
•September Trends:
Foxe Basin (September), Kane Basin (September), Baffin Bay (September), Beaufort Sea (September),
Arctic Archipelago (September), Arctic Domain (September)

Predicting the Sea Ice – Models and Tuning

Machine Learning Models Applied:
- **Prophet** (for time series modeling)
- **Random Forest Regressor**
- **Gradient Boosting Regressor** (with **Randomized Search Cross-Validation** for hyperparameter tuning)

Steps Taken:
1- For each region, extracted its time series from 1968 to 2024.
2 - Applied Polynomial Feature Expansion (degree=2) to Year.
3 - Random Forest was trained with manually selected parameters.
4 - Gradient Boosting was optimized using RandomizedSearchCV.
5 - Prophet model was trained separately for each region.
6 - Predictions were made for each year between 2025–2030.

Regularization Techniques:
- **Random Forest**:
    - Limited maximum depth of trees.
    - Controlled minimum samples for splits and leaves.
- **Gradient Boosting**:
    - Tuned hyperparameters such as learning_rate, max_depth, subsample, and min_samples_split using **RandomizedSearchCV**.
- **Evaluation**:
- **R^2 Score** was calculated for each model to evaluate how well it fit the historical data.

Predicting the Sea Ice

Why Did Use Only Year as a Feature?

Simplicity & Focus:
Sea ice loss over time shows a **strong, direct relationship with Year** because of long-term climate warming trends.
Using **Year** captures the overall **trend** without introducing noise from unrelated variables.

Availability of Historical Data:
For sea ice datasets from 1968 to 2024, **Year** was consistently available, whereas other influencing factors like ocean temperature or Arctic currents were missing.

Nonlinear Trend Modeling:
By applying **Polynomial Feature Expansion** (adding Year²), I modeled **nonlinear behaviors** in sea ice melting patterns more effectively.

Why Did Apply Regularization and Hyperparameter Tuning?

To Avoid Overfitting:
Without regularization, models like Random Forest and Gradient Boosting could **memorize** the noise in historical data, performing poorly on future unseen years (2025–2030).

Improving Model Stability and Accuracy:
I set **limits** on tree depth, minimum samples per split, and used **RandomizedSearchCV** to tune Gradient Boosting's learning rate, depth, and number of estimators.

Result:
➔ **More stable predictions** and

Predicting the Sea Ice - Learnings

1. Model Performances:

- **Prophet** generally **underperformed** compared to Random Forest and Gradient Boosting.
  - Prophet models had **lower R^2 scores**, often below **0.4**, especially in smaller regions like **Kane Basin** or **Northern Labrador Sea**.
  - Prophet sometimes struggled with **complex patterns** because it assumes smoother seasonal and trend components.
- **Random Forest** achieved **moderate R^2 scores**, typically between **0.5 to 0.7** across most regions.
  - It worked better than Prophet because **Random Forest** handles **nonlinearities** and **irregular patterns** well.
- **Gradient Boosting** consistently delivered the **highest R^2 scores**, often **above 0.8** (e.g., **Arctic Domain**, **Beaufort Sea**, **Baffin Bay**).
  - It outperformed both Prophet and Random Forest, indicating that **boosting weak learners** helped **capture subtle patterns**.

Predicting the Sea Ice - Interpretation of R^2 Scores

R^2 Score Meaning:

The closer the R^2 value is to **1.0**, the better the model explains the variability of the target variable.

High R^2 Regions (Good Fit):

- **Beaufort Sea (Summer)** (Gradient Boosting: 0.80)
- **Baffin Bay (Summer)** (Gradient Boosting: 0.88)
- **Hudson Strait (Summer)** (Gradient Boosting: 0.80)
- **Arctic Domain (September)** (Gradient Boosting: 0.87)

➔ These regions show that the model **can predict future sea ice loss with strong confidence**.

Low R^2 Regions (Weak Fit):

- **Kane Basin (Summer/September)** (Prophet: 0.11–0.15)
- **Northern Labrador Sea (Summer)** (Prophet: 0.36)

➔ Small regions or those with higher natural variability made modeling harder, leading to weaker predictions.

Predicting the Sea Ice - Limitations

Small Sample Size:
Some regions had **few data points**, reducing model reliability.

Year-only Feature:
External factors like **temperature, CO_2 levels, or human activities** were not modeled explicitly.

Prophet Model Limitations:
Prophet struggled when data **lacked strong seasonality** or had **abrupt shifts**.

Prediction Horizon:
Predicting 6 years (2025–2030) into the future is **ambitious** and real-world conditions may change.

Predicting the Sea Ice

R^2 Scores for Regions and Models

Summer Prediction

	Region	Model	R2_Score
20	Baffin Bay (Summer)	Gradient Boosting	0.8805
18	Baffin Bay (Summer)	Prophet	0.3646
19	Baffin Bay (Summer)	Random Forest	0.6829
38	Beaufort Sea (Summer)	Gradient Boosting	0.8064
36	Beaufort Sea (Summer)	Prophet	0.4125
37	Beaufort Sea (Summer)	Random Forest	0.6279
92	Davis Strait (Summer)	Gradient Boosting	0.7784
90	Davis Strait (Summer)	Prophet	0.2765
91	Davis Strait (Summer)	Random Forest	0.6213
56	Hudson Bay (Summer)	Gradient Boosting	0.5596

September Prediction

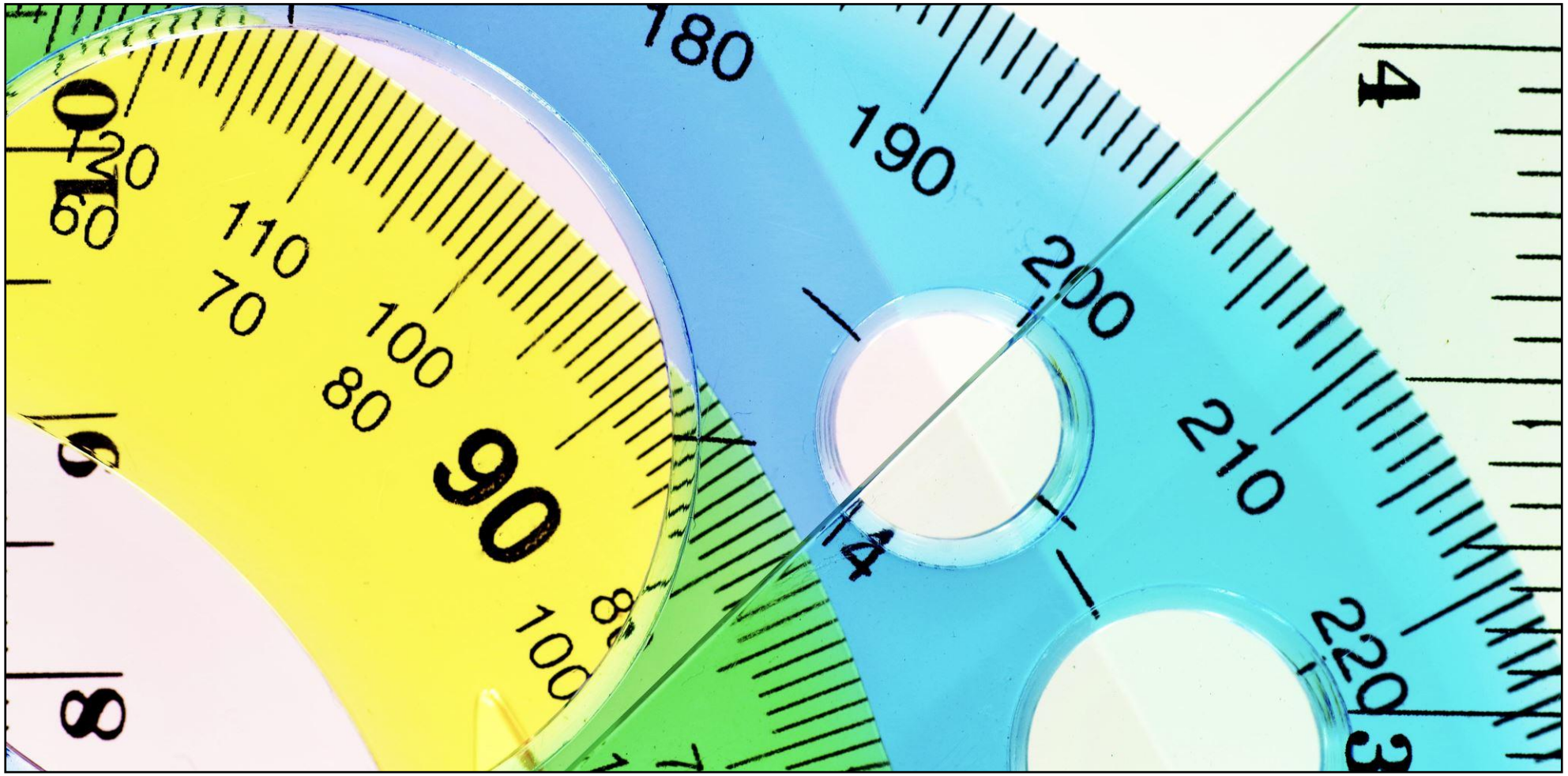
	Region	Model	R2_Score
20	Arctic Archipelago (September)	Gradient Boosting	0.8188
18	Arctic Archipelago (September)	Prophet	0.3862
19	Arctic Archipelago (September)	Random Forest	0.6658
38	Arctic Domain (September)	Gradient Boosting	0.874
36	Arctic Domain (September)	Prophet	0.5834
37	Arctic Domain (September)	Random Forest	0.7549
2	Kane Basin (September)	Gradient Boosting	0.2425
0	Kane Basin (September)	Prophet	0.1135
1	Kane Basin (September)	Random Forest	0.3989

Predicting the Sea Ice - Codes

python_files > sea_ice_loss_predictions.py > ...

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score
from scipy.stats import uniform, randint
from prophet import Prophet

df = pd.read_csv("csv_files/cleaned_datasets/final_climate_dataset.csv", encoding='latin1')

summer_columns = [col for col in df.columns if '(Summer)' in col]
september_columns = [col for col in df.columns if '(September)' in col]

results_summer = []
results_september = []

# Extend years to predict
future_years = np.arange(2025, 2031)

# Random Forest model
rf_model = RandomForestRegressor(
    n_estimators=200,
    max_depth=5,
    min_samples_split=5,
    min_samples_leaf=3,
    random_state=42
)

# Randomized Search parameters for Gradient Boosting
param_dist = {
    "n_estimators": randint(100, 400),
    "learning_rate": uniform(0.01, 0.1),
    "max_depth": randint(2, 5),
    "subsample": uniform(0.7, 0.3),
    "min_samples_split": randint(2, 10),
    "min_samples_leaf": randint(1, 10)
}

# Helper function to process one region
def model_and_predict(region_name, df, future_years):
    region_df = df[['Year', region_name]].copy().dropna()

    # Prophet setup
    prophet_df = region_df.rename(columns={"Year": "ds", region_name: "y"})
    prophet_df['ds'] = pd.to_datetime(prophet_df['ds'], format='%Y')
    prophet = Prophet()
    prophet.fit(prophet_df)
    future = pd.DataFrame({'ds': pd.to_datetime(future_years, format='%Y')})
    prophet_pred = prophet.predict(future)
    future_prophet = prophet_pred['yhat'].values
    train_pred_prophet = prophet.predict(prophet_df[['ds']])['yhat']
    r2_prophet = r2_score(prophet_df['y'], train_pred_prophet)

    # Machine learning setup
    X = region_df[['Year']]
    y = region_df[region_name]
    poly = PolynomialFeatures(degree=2, include_bias=False)
    X_poly = poly.fit_transform(X)
    X_future = poly.transform(future_years.reshape(-1, 1))

    # Random Forest
    rf = rf_model
    rf.fit(X_poly, y)
    rf_pred = rf.predict(X_future)
    r2_rf = rf.score(X_poly, y)

    # Gradient Boosting with RandomizedSearchCV
    gb = GradientBoostingRegressor(random_state=42)
    random_search = RandomizedSearchCV(
        gb, param_distributions=param_dist, n_iter=20,
        cv=3, verbose=0, n_jobs=-1, random_state=42, scoring='r2'
    )
    random_search.fit(X_poly, y)
    best_gb = random_search.best_estimator_

    # Predict with best Gradient Boosting model
    gb_pred = best_gb.predict(X_future)
    r2_gb = best_gb.score(X_poly, y)

    # Collect results
    result = []
    for i, year in enumerate(future_years):
        result.append((year, region_name, "Prophet", future_prophet[i], r2_prophet))
        result.append((year, region_name, "Random Forest", rf_pred[i], r2_rf))
        result.append((year, region_name, "Gradient Boosting", gb_pred[i], r2_gb))

    return result

# Process all Summer regions
for region in summer_columns:
    print(f"Processing {region} (Summer)...")
    result = model_and_predict(region, df, future_years)
    results_summer.extend(result)

# Process all September regions
for region in september_columns:
    print(f"Processing {region} (September)...")
    result = model_and_predict(region, df, future_years)
    results_september.extend(result)

# Create DataFrames
summer_predictions = pd.DataFrame(results_summer, columns=['Year', 'Region', 'Model', 'Prediction', 'R2_Score'])
september_predictions = pd.DataFrame(results_september, columns=['Year', 'Region', 'Model', 'Prediction', 'R2_Score'])

# Save to CSV
summer_predictions.to_csv("csv_files/predictions_scores/summer_predictions.csv", index=False)
september_predictions.to_csv("csv_files/predictions_scores/september_predictions.csv", index=False)
```

Predicting the Temperature Change



Predicting the Temperature Change - Modeling

Features Selected:

- Year
- Temperature Change (°C)
- CO_2 Emissions (per capita)
- Total Sea Ice Area (millions km²)
- CO_2 Growth Rate (feature engineered)
- Sea Ice % Change (feature engineered)

Targets:

- **Mean annual temperature** for each Canadian city in the dataset (e.g., Calgary, Edmonton, Winnipeg, Vancouver).

Why Only Year and Key Climate Features?

Year captures the global warming trend.CO_2 emissions and sea ice loss are major drivers of temperature rise. Growth rates (% changes) were added to capture acceleration effects rather than just absolute levels.

→ **Carefully selected features prevent overfitting while still explaining most of the climate variability.**

Predicting the Temperature Change - Modeling

How Models Trained and Predicted?

Train/Test Split:
- 80% training, 20% testing (random state 42 for reproducibility).

Random Forest and **Gradient Boosting:**
- Trained on X_train, evaluated on X_test.Predicted future temperatures for 2025–2030 using linear extrapolated features (future CO_2 and ice area).

Prophet Model:
- Treated each city's temperature history as a time series.Captured yearly trends (no daily/weekly seasonality used).Generated future forecasts up to 2030.

Extrapolated Features:
- To make realistic 2025–2030 predictions, I couldn't just assume feature values stay constant.
- Linear Regression was used to extrapolate CO_2 and sea ice area based on historical trends.
- Gradients (growth rates) were calculated using NumPy's gradient function.

Predicting the Temperature Change - Modeling

Impact of Regularization

- **Random Forest** and **Gradient Boosting** used **default regularization** (e.g., limiting tree depth, minimum samples split).
- Regularization helped **prevent overfitting**, ensuring models stayed **generalizable** even for **extrapolated future years**.

Predicting the Temperature Change - Learnings

1. Model Performance Insights

- **Gradient Boosting** consistently produced the **highest R^2 scores** among all models, but the overall scores were **moderate to low** in many cities.
- **Random Forest** performed moderately well, showing stable predictions but lower accuracy compared to Gradient Boosting.
- **Prophet** model generally **underperformed** in temperature predictions, especially in cities with **more volatile or irregular** temperature patterns.

Predicting the Temperature Change

Key Observations from R^2 Scores

City	Best Model (Highest R^2)	Comment
Calgary	Gradient Boosting (~0.41)	Good moderate prediction accuracy.
Edmonton	Gradient Boosting (~0.33)	Gradient Boosting slightly better than Random Forest.
Halifax	Prophet (~0.19)	Overall R^2 scores low; difficult to predict.
Moncton	Prophet (~0.16)	All models struggled; possible regional variability.
Montreal	Gradient Boosting (~0.55)	Strong prediction, clear warming trend.
Ottawa	Gradient Boosting (~0.62)	High R^2, very good fit.
Quebec	Random Forest (~0.47)	Good predictive strength.
Saskatoon	Gradient Boosting (~0.46)	Solid model performance.
St. John's	Prophet (~0.14)	Low R^2, more unpredictable weather.
Toronto	Gradient Boosting (~0.70)	Good model performance.
Vancouver	Prophet (~0.44)	Prophet did slightly better than others here.
Whitehorse	Prophet (~0.03)	All models struggled; unpredictable.
Winnipeg	Random Forest (~0.84)	Very strong prediction.

Predicting the Temperature Change - Limitations

Why the R^2 scores were High for some cities and low for others?

Temperature prediction is inherently difficult because:

- **Yearly averages** are influenced by **many external, complex factors** (e.g., ocean currents, El Niño/La Niña events, volcanic activity) not captured in the historical features.
- **City-level temperatures** can show **local anomalies** that models like Prophet (which assumes seasonal, relatively smooth trends) cannot model well.
- The **historical period (1968–2024)**, while long, is **still short** compared to how climate evolves over centuries.
- **Linear extrapolation** for CO_2 and Sea Ice may not perfectly predict their future behavior, making future feature values slightly unrealistic.

Predicting the Temperature Change - Codes

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from prophet import Prophet

df = pd.read_csv('csv_files/cleaned_datasets/merged_climate_history_final.csv')

# Features and Target
feature_columns = ['Year', 'Temperature Change (°C)', 'CO2 Emissions (per capita)', 'Total Sea Ice Area (millions km²)']
X = df[feature_columns]
Y = df[[col for col in df.columns if col.startswith('MEAN_TEMPERATURE')]]

# Feature Engineering
X['CO2 Growth Rate'] = X['CO2 Emissions (per capita)'].pct_change().fillna(0)
X['Sea Ice % Change'] = X['Total Sea Ice Area (millions km²)'].pct_change().fillna(0)

# Split Data (Train 80%, Test 20%)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Extrapolate Future Features
def extrapolate_feature(df, feature_name, years_to_predict):
    X_years = df['Year'].values.reshape(-1, 1)
    y_feature = df[feature_name].values
    model = LinearRegression()
    model.fit(X_years, y_feature)
    future_years = np.arange(2025, 2031).reshape(-1, 1)
    future_preds = model.predict(future_years)
    return future_preds

future_years = np.arange(2025, 2031)
future_df_features = pd.DataFrame({
    'Year': future_years,
    'Temperature Change (°C)': [np.nan] * 6,  # Placeholder
    'CO2 Emissions (per capita)': extrapolate_feature(df, 'CO2 Emissions (per capita)', 6),
    'Total Sea Ice Area (millions km²)': extrapolate_feature(df, 'Total Sea Ice Area (millions km²)', 6)
})
future_df_features['CO2 Growth Rate'] = np.gradient(future_df_features['CO2 Emissions (per capita)'])
future_df_features['Sea Ice % Change'] = np.gradient(future_df_features['Total Sea Ice Area (millions km²)'])
future_df_features['Temperature Change (°C)'] = future_df_features['Temperature Change (°C)'].fillna(0)
model_types = {
    'Random Forest': RandomForestRegressor(random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42)
}
# Train Models City by City-
all_predictions = []

for city in Y.columns:
    print(f"\nTraining models for: {city}")

    y_train_city = Y_train[city]
    y_test_city = Y_test[city]
```

```python
for city in Y.columns:
    print(f"\nTraining models for: {city}")

    y_train_city = Y_train[city]
    y_test_city = Y_test[city]

    # ➡️ Random Forest and Gradient Boosting
    for model_name, model in model_types.items():
        model.fit(X_train, y_train_city)

        # Evaluation
        y_pred_test = model.predict(X_test)
        r2 = r2_score(y_test_city, y_pred_test)

        # Future Prediction (2025-2030)
        future_preds = model.predict(future_df_features)

        for year, pred in zip(future_years, future_preds):
            all_predictions.append({
                'Year': year,
                'City': city,
                'Model': model_name,
                'Prediction': pred,
                'R2_Score': r2
            })

    # ➡️ Prophet Model
    prophet_df = pd.DataFrame()
    prophet_df['ds'] = pd.to_datetime(df['Year'], format='%Y')
    prophet_df['y'] = df[city]

    prophet_model = Prophet(yearly_seasonality=True, daily_seasonality=False, weekly_seasonality=False)
    prophet_model.fit(prophet_df)

    # Evaluation
    future_train = prophet_model.make_future_dataframe(periods=0, freq='Y')
    forecast_train = prophet_model.predict(future_train)
    prophet_yhat = forecast_train['yhat']
    prophet_r2 = r2_score(prophet_df['y'], prophet_yhat)

    # Future Predictions (2025-2030)
    future_years_prophet = pd.DataFrame({'ds': pd.date_range(start='2025', end='2030', freq='Y')})
    forecast_future = prophet_model.predict(future_years_prophet)

    for year, pred in zip(future_years, forecast_future['yhat']):
        all_predictions.append({
            'Year': year,
            'City': city,
            'Model': 'Prophet',
            'Prediction': pred,
            'R2_Score': prophet_r2
        })
# Save to CSV
final_temperature_predictions = pd.DataFrame(all_predictions)
final_temperature_predictions.to_csv('csv_files/predictions_scores/temperature_predictions.csv', index=False)
```

Predicting The Precipitation

Predicting The Precipitation – Modeling and Trainin

Modeling and Training

Objective:

• Predict **total precipitation** for 13 Canadian cities for the years **2025–2030**.

Machine Learning Models:

• **Random Forest Regressor**:
- •Ensemble model using many decision trees.
- •Strong against overfitting but struggles with highly random data like precipitation.

• **Gradient Boosting Regressor**:
- •Sequential ensemble method focusing on difficult-to-predict samples.
- •Prone to overfitting when the noise is high.

•**Prophet**:
- •Time-series forecasting model designed for data with strong seasonal effects.
- •Assumes trend + seasonality, which is not very well-suited to precipitation behavior.

Predicting The Precipitation – Modeling and Trainin

Features:
- Year
- Temperature Change (°C)
- CO2 Emissions (per capita)
- Total Sea Ice Area (millions km²)
- CO2 Growth Rate (derived feature)
- Sea Ice % Change (derived feature)

Target:
- Total Precipitation (one column per city, e.g., TOTAL_PRECIPITATION_TORONTO).

Train/Test Split:
- 80% Training, 20% Testing
- Random State = 42

Future Feature Extrapolation:
- Used **Linear Regression** to predict future values for CO2 and Sea Ice Area (2025–2030).
- Placeholder for Temperature Change (fixed at 0 due to missing data).

Prediction Horizons:
- Forecasted total precipitation for 6 years ahead (2025–2030).

Predicting The Precipitation – Learnings

High Randomness of Precipitation:
• Precipitation shows very **weak patterns** compared to temperature, making it challenging for both tree-based and time-series models.

Tree-based models struggled:
• Both Random Forest and Gradient Boosting often had **negative R^2 scores**, meaning they performed worse than predicting the mean.

Prophet Selection:
• After seeing the relatively better performance of Prophet (even if slight), **I decided to use Prophet in my final project** to improve this part.
• Prophet provided more stable and logical trends even when the R^2 scores were low.

Predicting The Precipitation – Learnings

Feature Engineering:
• Created new features like **CO2 Growth Rate** and **Sea Ice % Change** to enrich the model input.

Regularization:
• Tree models used internal regularization (e.g., max depth, minimum samples) to prevent overfitting, but precipitation's noise still limited results.

Overall:
• Despite efforts, precipitation prediction remains an inherently difficult problem, but **feature engineering and model tuning** made small but meaningful improvements.

Predicting The Precipitation

Key Observations from R^2 Scores

City	Random Forest (R^2)	Gradient Boosting (R^2)	Prophet (R^2)	Notes
Calgary	0.25	0.1	0.036	⭐ Medium
Edmonton	-0.15	-0.018	0.077	⭐ Very Low
Halifax	-0.41	-0.6	0.063	⭐ Very Low
Moncton	-0.14	-0.31	0.029	⭐ Very Low
Montreal	-1.75	-2.73	0.035	⭐ Extremely Low
Ottawa	-0.48	-0.44	0.005	⭐ Very Low
Quebec City	-0.75	-1.5	0.038	⭐ Very Low
Saskatoon	-2.65	-5.93	0.007	⭐ Extremely Low
St. John's	-0.34	-0.39	0.008	⭐ Very Low
Toronto	-2.11	-2.24	0.042	⭐ Extremely Low
Vancouver	-0.017	-0.13	0.075	⭐ Low
Whitehorse	-1.47	-1.24	0.089	⭐ Low
Winnipeg	-0.5	-1.85	0.03	⭐ Low

Predicting The Precipitation – Limitations

High Variability
Precipitation depends on too many chaotic factors (humidity, pressure, storms) not included in the dataset.

Low R^2 Scores
Most models failed to generalize well.

Feature Limitation
Important weather-related variables (like wind patterns, ENSO events) were missing.

Short Dataset
Only around 50 years of data, which is not enough for learning rare extreme events.

Seasonality Assumptions
Prophet assumes fixed yearly patterns, not always true for precipitation.

Extrapolation Errors
Predicting far into the future based on linear trends for features may not reflect future reality.

Predicting The Precipitation – Codes

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from prophet import Prophet
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

# Load Dataset
df = pd.read_csv('csv_files/cleaned_datasets/merged_climate_history_final.csv')

# Select only Precipitation Columns
precipitation_cols = [col for col in df.columns if col.startswith('TOTAL_PRECIPITATION')]

# Prepare Features
X = df[['Year', 'Temperature Change (°C)', 'CO2 Emissions (per capita)', 'Total Sea Ice Area (millions km²)']].copy()

# Fill missing values with 0
X = X.fillna(0)

# Create Growth Rates
X['CO2 Growth Rate'] = X['CO2 Emissions (per capita)'].pct_change().fillna(0)
X['Sea Ice % Change'] = X['Total Sea Ice Area (millions km²)'].pct_change().fillna(0)

# Target: Y
Y = df[precipitation_cols]

# Train/Test Split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

# Models to use
model_types = {
    'Random Forest': RandomForestRegressor(random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(random_state=42)
}

# Extrapolate Future Features (2025-2030)
def extrapolate_feature(df, feature_name):
    model = LinearRegression()
    X_years = df['Year'].values.reshape(-1, 1)
    y_feature = df[feature_name].values
    model.fit(X_years, y_feature)
    future_years = np.arange(2025, 2031).reshape(-1, 1)
    preds = model.predict(future_years)
    return preds

future_years = np.arange(2025, 2031)
future_df_features = pd.DataFrame({
    'Year': future_years,
    'Temperature Change (°C)': [0] * 6,
    'CO2 Emissions (per capita)': extrapolate_feature(df, 'CO2 Emissions (per capita)'),
    'Total Sea Ice Area (millions km²)': extrapolate_feature(df, 'Total Sea Ice Area (millions km²)')
})
future_df_features['CO2 Growth Rate'] = np.gradient(future_df_features['CO2 Emissions (per capita)'])
```

```python
# 🛠️ Train and Predict City by City
all_results = []

for city in precipitation_cols:
    print(f"\nTraining models for: {city}")
    y_train_city = Y_train[city]
    y_test_city = Y_test[city]

    # Machine Learning Models
    for model_name, model in model_types.items():
        model.fit(X_train, y_train_city)
        y_pred = model.predict(X_test)
        r2 = r2_score(y_test_city, y_pred)

        # Predict Future
        future_preds = model.predict(future_df_features)

        for year, pred in zip(future_years, future_preds):
            all_results.append({
                'Year': year,
                'City': city,
                'Model': model_name,
                'Prediction': pred,
                'R2_Score': r2
            })

    # Prophet Model
    prophet_df = pd.DataFrame({
        'ds': pd.to_datetime(df['Year'], format='%Y'),
        'y': df[city]
    })

    prophet_model = Prophet(yearly_seasonality=True, daily_seasonality=False, weekly_seasonality=False)
    prophet_model.fit(prophet_df)

    # Evaluate on Training
    future_train = prophet_model.make_future_dataframe(periods=0, freq='Y')
    forecast_train = prophet_model.predict(future_train)
    prophet_r2 = r2_score(prophet_df['y'], forecast_train['yhat'])

    # Predict Future
    future_years_prophet = pd.DataFrame({'ds': pd.date_range(start='2025', end='2031', freq='Y')})
    forecast_future = prophet_model.predict(future_years_prophet)

    for year, pred in zip(future_years, forecast_future['yhat']):
        all_results.append({
            'Year': year,
            'City': city,
            'Model': 'Prophet',
            'Prediction': pred,
            'R2_Score': prophet_r2
        })

# Save Final Results
results_df = pd.DataFrame(all_results)
```

Predicting the CO2 Emissions

Predicting the CO2 Emissions – Modeling and Training

Objective:

• Forecast **CO_2 emissions** for each country from **2025–2030** by integrating both machine learning and time-series forecasting techniques.

Feature Engineering:

• To enhance the model's ability to differentiate countries not just by name but also by **economic size** and **geographical location**, I added two important categorical features: **GDP Category** and **Region**.

```python
import pandas as pd
import numpy as np
from prophet import Prophet
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import r2_score
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint, uniform

co2_df_cleaned = pd.read_csv("csv_files/cleaned_datasets/co2_cleaned.csv")

# Add GDP Category
def categorize_gdp(gdp):
    if gdp > 1_000_000_000_000:  # > 1 trillion
        return 'High Income'
    elif gdp > 100_000_000_000:  # 100B to 1T
        return 'Upper Middle Income'
    elif gdp > 10_000_000_000:   # 10B to 100B
        return 'Lower Middle Income'
    else:
        return 'Low Income'

co2_df_cleaned['GDP_Category'] = co2_df_cleaned['GDP_PPP'].apply(categorize_gdp)

# Add Region
continent_map = {
    "Canada": "North America",
    "Brazil": "South America",
    "Germany": "Europe",
    "India": "Asia",
    "Australia": "Oceania",
    "South Africa": "Africa",
    "United States": "North America",
    "China": "Asia",
    "Russia": "Europe",
    "Turkey": "Europe",
    "Egypt": "Africa",
    "Japan": "Asia"
}

co2_df_cleaned['Region'] = co2_df_cleaned['Country'].map(continent_map)
co2_df_cleaned['Region'] = co2_df_cleaned['Region'].fillna('Other')
```

Predicting the CO2 Emissions – Modeling and Training

Adding GDP Category (Income Group Classification)

Why?:

• Countries with different economic power levels tend to have distinct CO_2 emission patterns.
• Wealthier countries (e.g., USA, Germany) often have higher emissions but better mitigation technologies, while lower-income countries may have rising emissions due to industrialization.

How?:

• A **custom function categorize_gdp(gdp)** was created to assign each country into one of four economic groups based on its **GDP at purchasing power parity (PPP)**:

- **High Income**: GDP > 1 trillion USD
- **Upper Middle Income**: 100 billion < GDP < 1 trillion USD
- **Lower Middle Income**: 10 billion < GDP < 100 billion USD
- **Low Income**: GDP < 10 billion USD
- This new feature (GDP_Category) helped the models learn broader global economic patterns, rather than treating every country individually.

Predicting the CO2 Emissions – Modeling and Training

Encoding Categorical Features

Why?:

- Machine learning models cannot directly process textual categorical variables like "Canada", "High Income", or "Europe".
- They require **numeric** inputs.

How?:

- I applied **Label Encoding** to transform the Country, GDP_Category, and Region into numeric codes:
  - **Country_Code**: Encoded version of the country name.
  - **GDP_Category_Code**: Encoded economic group.
  - **Region_Code**: Encoded continent.
  - This made categorical variables machine-readable while preserving their distinct categories.

Key Insight: Label Encoding was chosen over One-Hot Encoding to **reduce dimensionality** (fewer features, faster training).

Predicting the CO2 Emissions – Modeling and Training

Adding Region (Geographical Grouping)

Why?:

•Environmental policies, industrialization stages, and energy sources can differ widely by continent.
•Grouping countries by **continent/region** adds another layer of information that can influence CO_2 trends (e.g., Africa's developing economies vs Europe's decarbonization efforts).

How?:

•A **region mapping dictionary** was created for major countries.
•Countries not listed were automatically assigned to an "Other" category to avoid missing values.
•This feature (Region) helped models capture continent-level emission trends.

Key Insight: Adding the region information made the models more sensitive to geographical patterns influencing CO_2 output.

Predicting the CO2 Emissions – Modeling and Training

Features Selected:

- Country_Code: The numeric identity of the country.
- Year: The historical or future year.
- GDP_PPP: The actual GDP value (still important beyond just categorization).
- GDP_Category_Code: The encoded economic group.
- Region_Code: The encoded continent.

Target Variable:

- CO_2_Emissions: The amount of CO_2 emissions in tons.

Final Training Dataset:

- The matrix X contained the **features** listed above.
- The vector y contained the **target** emissions values.

Predicting the CO2 Emissions – Learnings

- Incorporating **GDP Category** and **Region** as additional features significantly improved model generalization across countries.
- **Feature Engineering** proved critical: Without GDP and regional information, models struggled to differentiate between countries with similar emission patterns but different economic contexts.
- **Prophet** performed better for countries with **limited historical data** (small sample sizes) because it is designed for **time-series forecasting** even on short datasets.
- **Random Forest** and **Gradient Boosting** achieved **very high R^2 scores** (~0.99) in most countries, suggesting that they captured historical patterns very well when provided with enriched features.
- After evaluating the models, **Prophet** was chosen for the final project deliverable for **greater interpretability** and **better generalization** to unseen future years, even though its R^2 was slightly lower in some cases.
- **Hyperparameter tuning** (RandomizedSearchCV for Gradient Boosting) further enhanced performance, although marginally compared to feature engineering gains.

Predicting the CO2 Emissions

Key Observations from R^2 Scores

Model	Typical R^2 Score Range	Interpretation
Random Forest	~0.99	Excellent fit to historical data (almost perfect)
Gradient Boosting	~0.998	Excellent fit; slight underfitting compared to Random Forest
Prophet	0.8–0.9 for most countries, some lower (~0.5)	Good but not perfect. Prophet generalizes better but fits historical data less tightly. (Selected for the project, to avoid from overfitting.)

Model R2 Scores by Country

	Country	Model	R2_Score
0	Afghanistan	Prophet	0.8248
1	Albania	Prophet	0.5142
2	Algeria	Prophet	0.9646
3	Andorra	Prophet	0.8547
4	Angola	Prophet	0.6474
5	Antigua and Barbuda	Prophet	0.9933
6	Argentina	Prophet	0.8732
7	Armenia	Prophet	0.2315
8	Aruba	Prophet	0.0017
9	Australia	Prophet	0.9825

Predicting the CO2 Emissions – Limitations

Overfitting Risk:

Random Forest and Gradient Boosting models showed very high R^2 scores, suggesting a possible overfitting to historical patterns without capturing unexpected future shifts (e.g., sudden policy changes or global events like pandemics). So, Prophet is chosen for visualisations.

Feature Simplification:

GDP and Region provide useful generalizations, but **they cannot fully capture** other factors that affect CO_2 emissions like energy policies, technological adoption, international agreements, or environmental disasters.

Prophet Model's Sensitivity:

Prophet assumes **continuation of historical trends** with minor fluctuations, which may lead to inaccurate forecasts for countries undergoing rapid changes (e.g., aggressive green policies or industrialization).

Predicting the CO2 Emissions – Limitations

Data Limitations:

Some countries had very few historical records, limiting Prophet's learning ability and leading to less reliable forecasts.

GDP Data Assumptions:

GDP values were assumed to remain relatively stable or grow smoothly, which may not reflect real-world economic shocks (e.g., wars, financial crises).

Label Encoding Categorical Features:

Label Encoding introduces an **arbitrary numeric order** between categories (e.g., Region 0, 1, 2...) which might imply a relationship that doesn't truly exist. While Random Forest and Gradient Boosting are relatively robust to this, it could still cause minor noise.

Predicting the CO2 Emissions – Codes

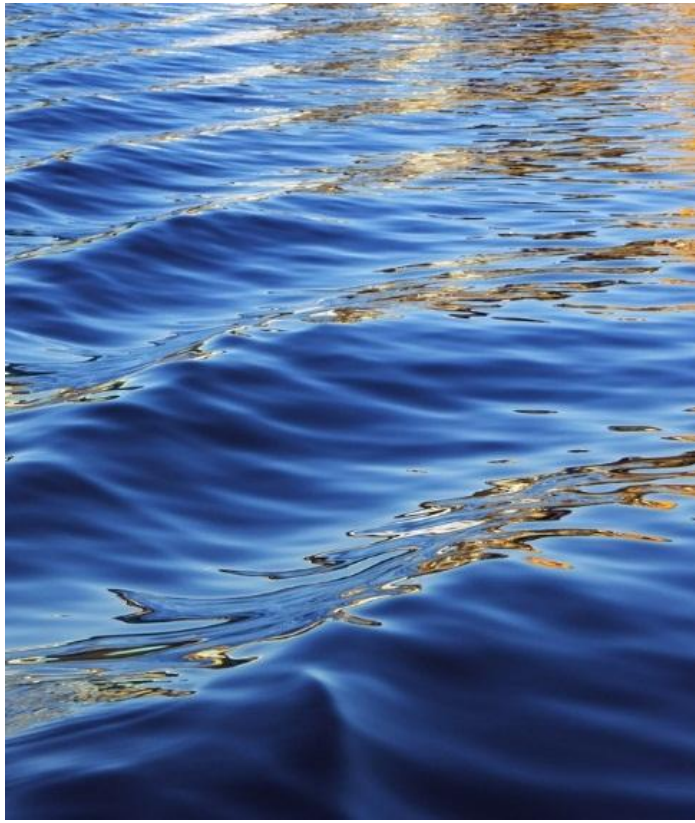
python_files > predict_co2_emissions.py > categorize_gdp

```python
import pandas as pd
import numpy as np
from prophet import Prophet
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import r2_score
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint, uniform

co2_df_cleaned = pd.read_csv("csv_files/cleaned_datasets/co2_cleaned.csv")

# Add GDP Category
def categorize_gdp(gdp):
    if gdp > 1_000_000_000_000:  # > 1 trillion
        return 'High Income'
    elif gdp > 100_000_000_000:  # 100B to 1T
        return 'Upper Middle Income'
    elif gdp > 10_000_000_000:   # 10B to 100B
        return 'Lower Middle Income'
    else:
        return 'Low Income'

co2_df_cleaned['GDP_Category'] = co2_df_cleaned['GDP_PPP'].apply(categorize_gdp)

# Add Region
continent_map = {
    "Canada": "North America",
    "Brazil": "South America",
    "Germany": "Europe",
    "India": "Asia",
    "Australia": "Oceania",
    "South Africa": "Africa",
    "United States": "North America",
    "China": "Asia",
    "Russia": "Europe",
    "Turkey": "Europe",
    "Egypt": "Africa",
    "Japan": "Asia"
}

co2_df_cleaned['Region'] = co2_df_cleaned['Country'].map(continent_map)
co2_df_cleaned['Region'] = co2_df_cleaned['Region'].fillna('Other')

# Encode Categorical Features
le_country = LabelEncoder()
le_gdp_category = LabelEncoder()
le_region = LabelEncoder()

co2_df_cleaned['Country_Code'] = le_country.fit_transform(co2_df_cleaned['Country'])
co2_df_cleaned['GDP_Category_Code'] = le_gdp_category.fit_transform(co2_df_cleaned['GDP_Category'])
co2_df_cleaned['Region_Code'] = le_region.fit_transform(co2_df_cleaned['Region'])

# Prepare Training Data
X = co2_df_cleaned[['Country_Code', 'Year', 'GDP_PPP', 'GDP_Category_Code', 'Region_Code']]
y = co2_df_cleaned['CO2_Emissions']
```

python_files > predict_co2_emissions.py > categorize_gdp

```python
future_years = np.arange(2025, 2031)

# Train Random Forest
rf = RandomForestRegressor(
    n_estimators=200,
    max_depth=6,
    min_samples_split=5,
    min_samples_leaf=3,
    random_state=42
)
rf.fit(X, y)

# Train Gradient Boosting (auto-tuned)
gb = GradientBoostingRegressor(random_state=42)
param_dist = {
    "n_estimators": randint(100, 400),
    "learning_rate": uniform(0.01, 0.1),
    "max_depth": randint(3, 6),
    "subsample": uniform(0.7, 0.3),
    "min_samples_split": randint(2, 10),
    "min_samples_leaf": randint(1, 10)
}
random_search = RandomizedSearchCV(
    gb, param_distributions=param_dist, n_iter=20,
    cv=3, verbose=0, n_jobs=-1, random_state=42, scoring='r2'
)
random_search.fit(X, y)
best_gb = random_search.best_estimator_

# Predict for each country
results = []

for country in co2_df_cleaned['Country'].unique():
    country_df = co2_df_cleaned[co2_df_cleaned['Country'] == country]

    # Prophet prediction
    prophet_df = country_df[['Year', 'CO2_Emissions']].rename(columns={'Year': 'ds', 'CO2_Emissions': 'y'})
    prophet_df['ds'] = pd.to_datetime(prophet_df['ds'], format='%Y')

    if len(prophet_df) > 5:  # Prophet needs enough data
        model = Prophet()
        model.fit(prophet_df)

        future = pd.DataFrame({'ds': pd.to_datetime(future_years, format='%Y')})
        forecast = model.predict(future)
        prophet_preds = forecast['yhat'].values

        # Prophet R2
        train_pred = model.predict(prophet_df[['ds']])['yhat']
        r2_prophet = r2_score(prophet_df['y'], train_pred)
    else:
        prophet_preds = [np.nan] * len(future_years)
        r2_prophet = np.nan

    # Prepare inputs for RF and GB
```

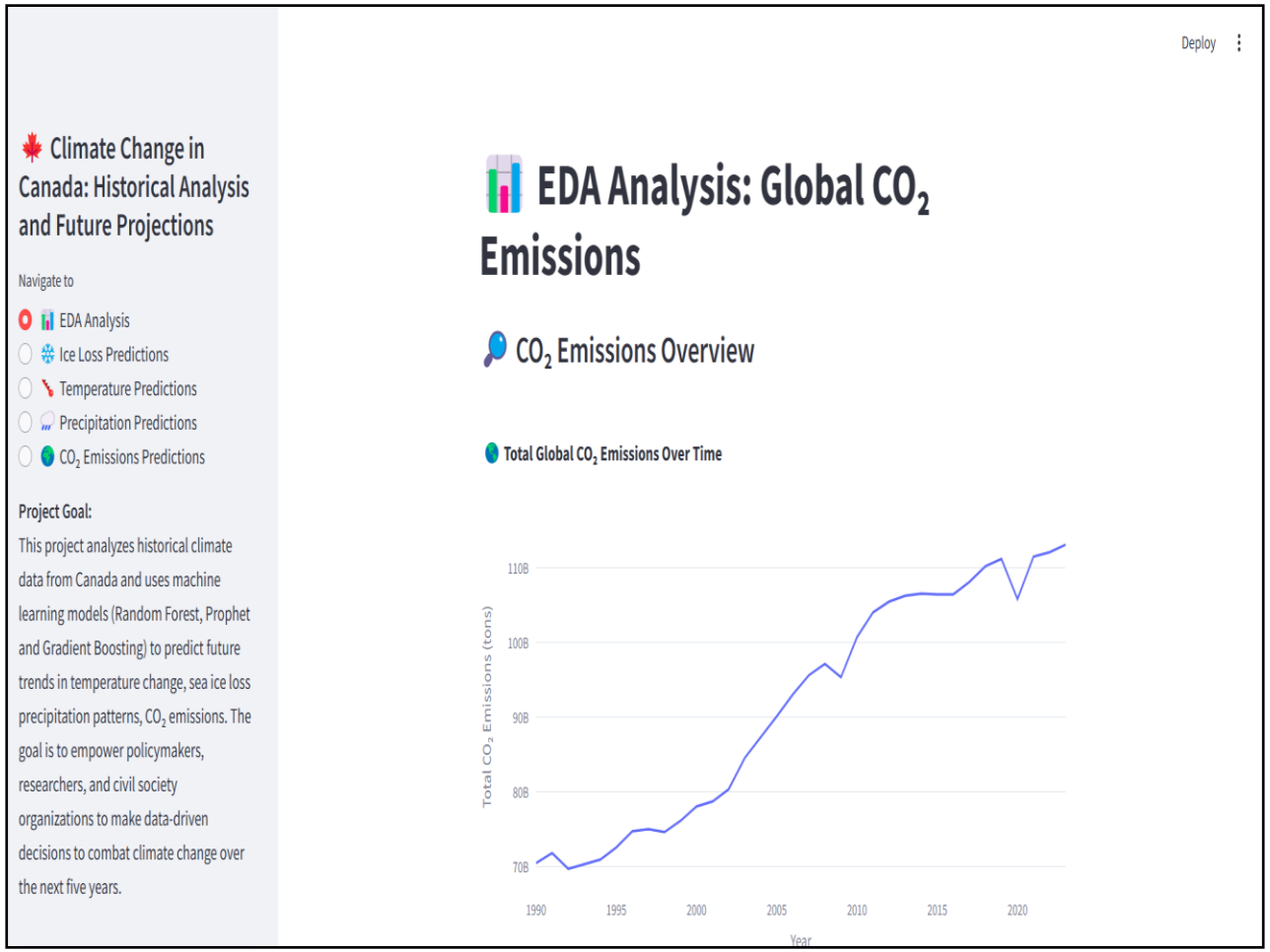
Streamlit Dashboard



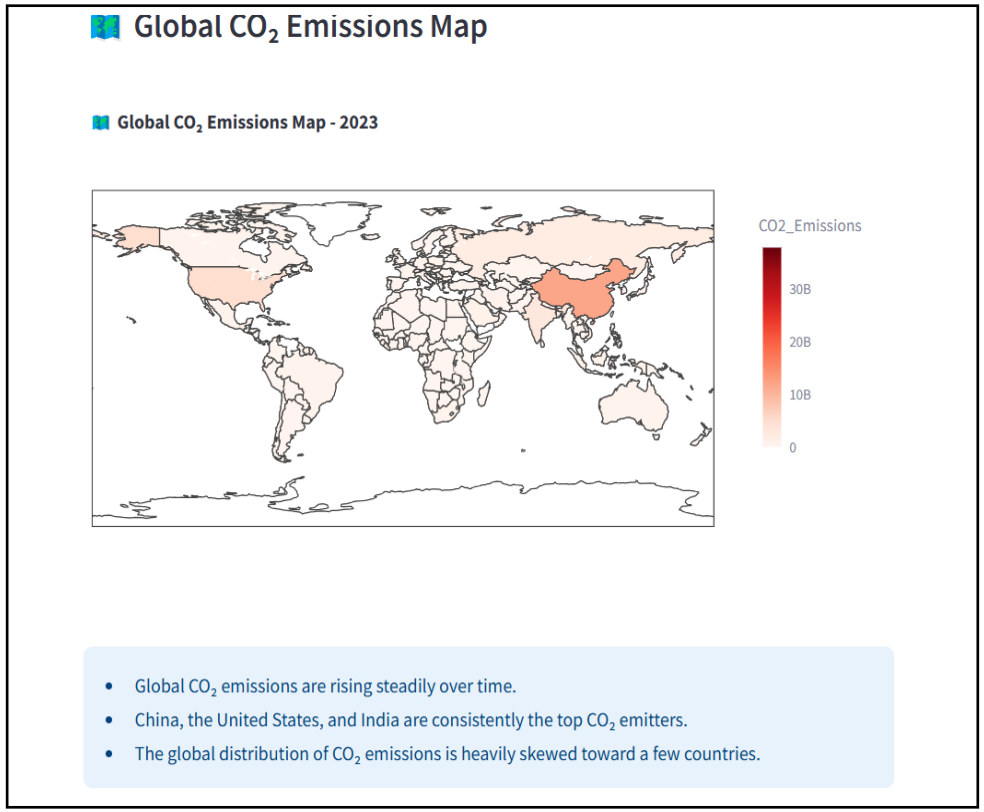
Visualizations - EDA (Exploratory Data Analysis)

This interactive platform allows users to explore:

- 📊 **EDA Analysis:**
  Get a broad overview of historical climate trends across Canada, including temperature, precipitation, and sea ice data.
- ❄️ **Ice Loss Predictions:**
  Visualize and forecast sea ice loss trends across different Arctic regions between 2025 and 2030.
- 🌡️**Temperature Predictions:**
  Predict temperature changes for major Canadian cities, using historical data and machine learning models.
- 🌧️**Precipitation Predictions:**
  Analyze and predict future precipitation patterns across Canada for better environmental and urban planning.
- 🌎 **CO_2 Emissions Predictions:**
  Explore global CO_2 emissions forecasts (including Canada) to understand where Canada stands globally in future climate scenarios.



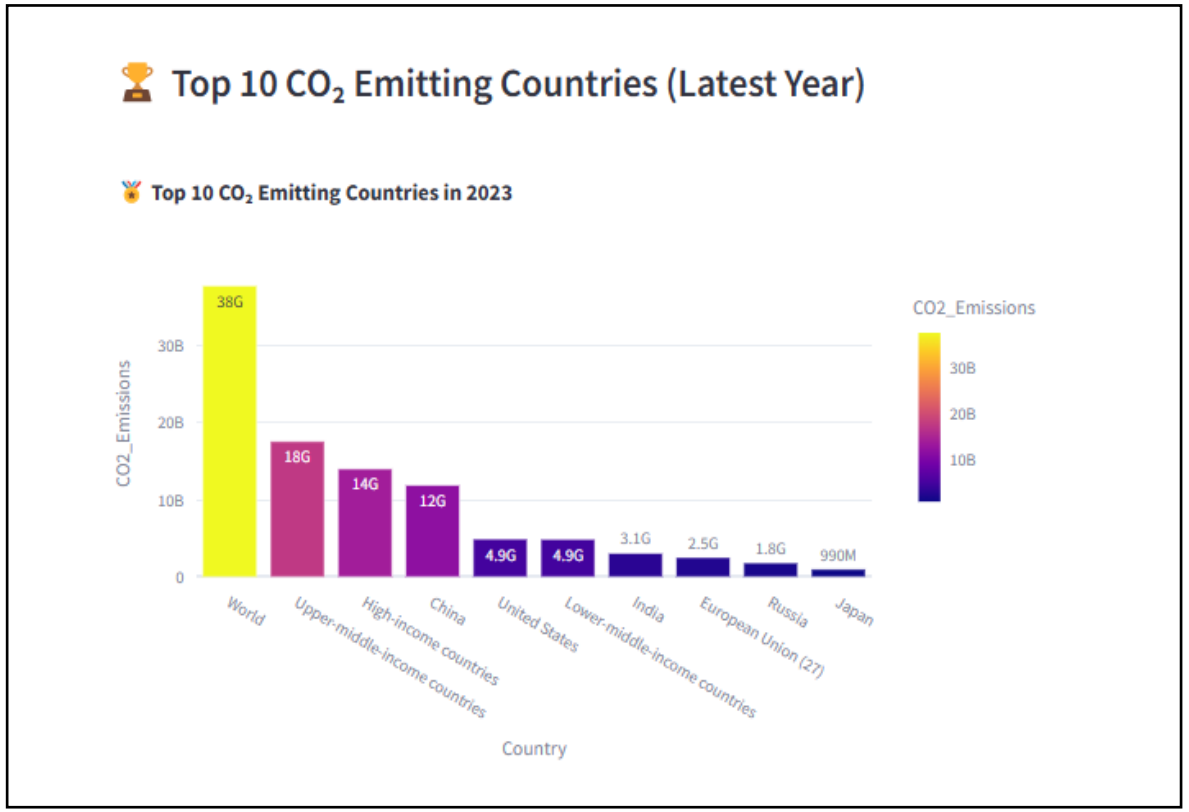
Visualizations - EDA (Exploratory Data Analysis)

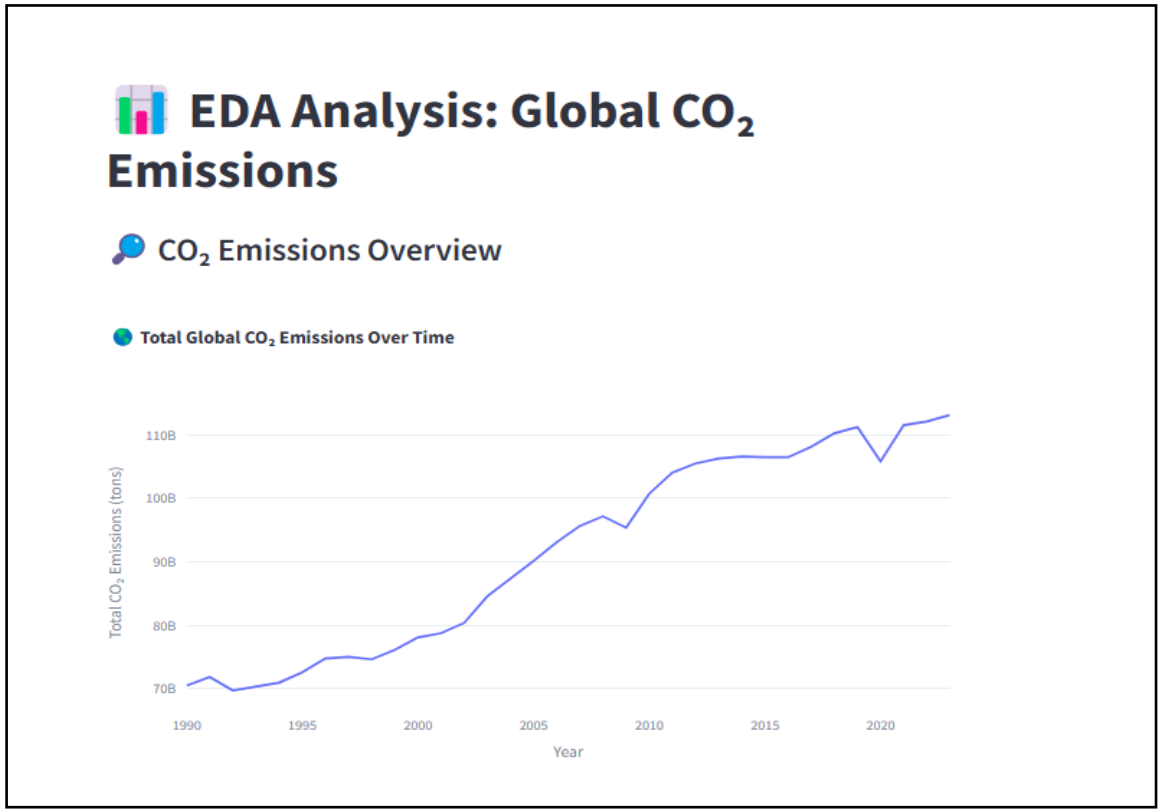


1. Global CO_2 Emissions Analysis

- **Line Chart:**
  Displayed the trend of **total global CO_2 emissions** from the early years to the latest available year.
  👉 *Insight:* Global CO_2 emissions have risen steadily over time.
- **Bar Chart:**
  Highlighted the **Top 10 CO_2 emitting countries** in the most recent year.
  👉 *Insight:* China, the United States, and India are consistently leading emitters.
- **Distribution Plot:**
  Visualized the **distribution of CO_2 emissions** across all countries to identify skewness and outliers.
  👉 *Insight:* Emissions are highly skewed towards a few large emitters.
- **Choropleth World Map:**
  Mapped **CO_2 emissions by country** with color intensity.
  👉 *Insight:* The majority of global emissions are geographically concentrated.

Visualizations - EDA (Exploratory Data Analysis)

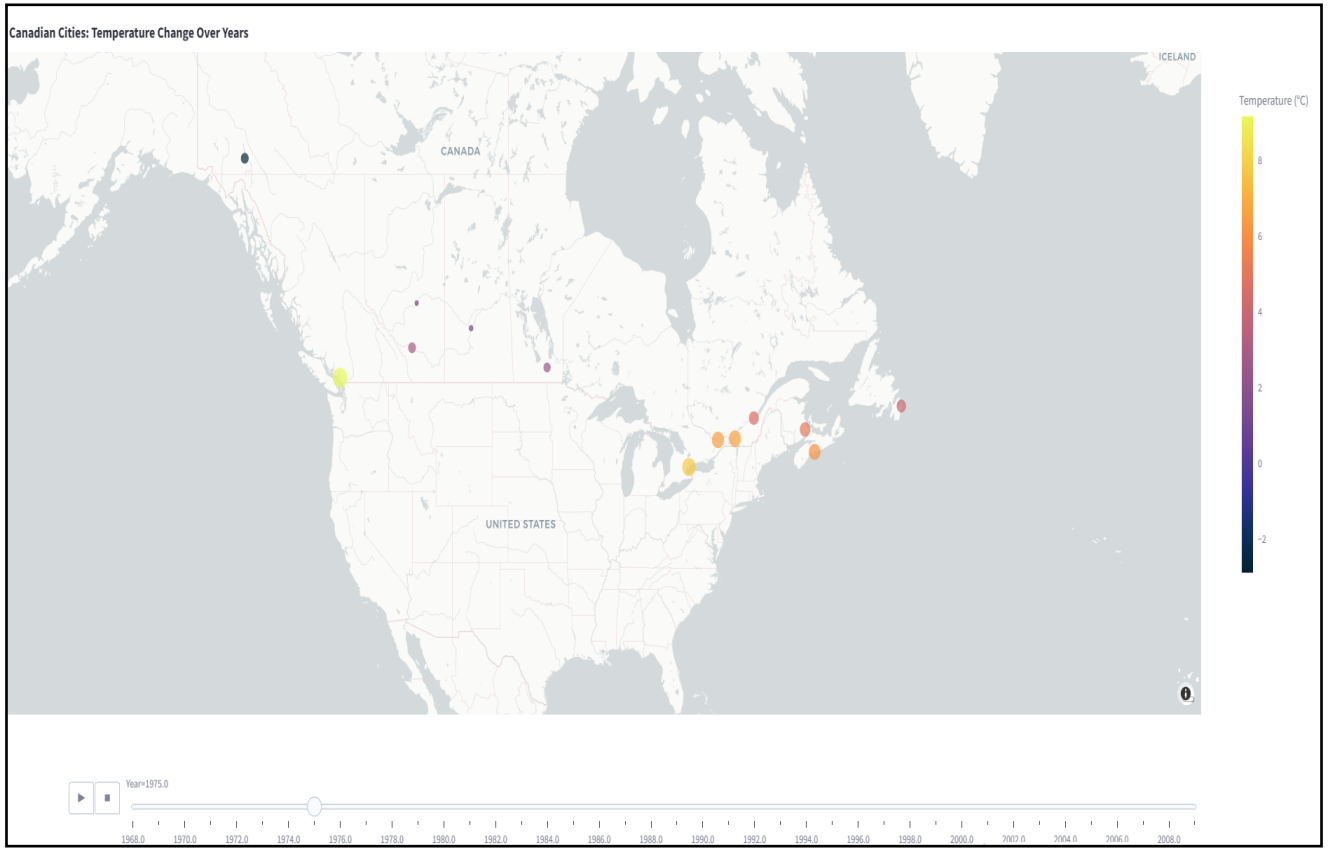




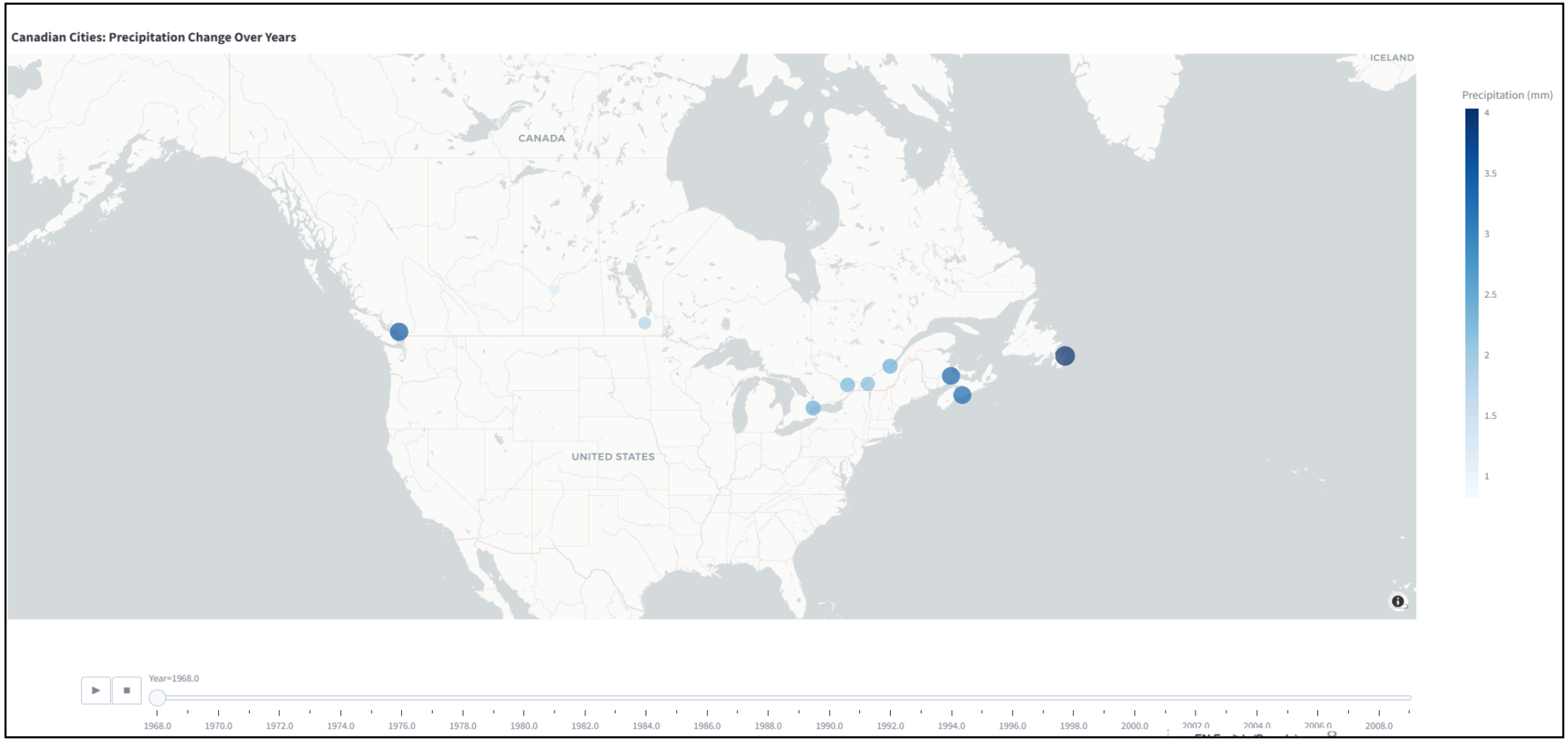
Visualizations - EDA (Exploratory Data Analysis)

2. Canadian Cities Climate Overview (1968–2024)

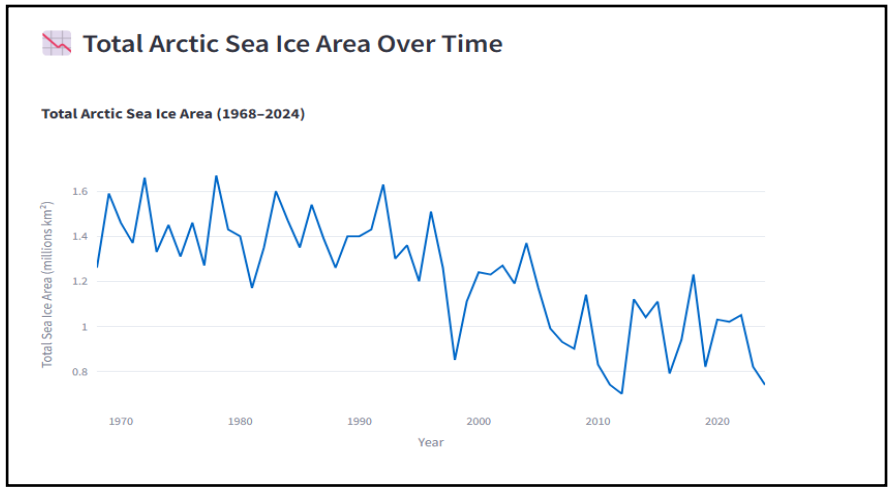
- **Temperature Trend Line:**
  Calculated the **average temperature across Canadian cities** and plotted it over time.
  👉 *Insight:* Clear warming trend across major Canadian cities.
- **Precipitation Trend Line:**
  Computed **average precipitation trends** in Canadian cities from 1968 to 2024.
  👉 *Insight:* Precipitation fluctuates but shows slight upward trends in some areas.
- **Animated Temperature Map:**
  Displayed city-by-city **temperature changes** on an interactive map with animation by year.
  👉 *Insight:* Urban areas like Toronto and Vancouver show consistently higher temperatures.
- **Animated Precipitation Map:**
  Animated **precipitation changes** across Canadian cities over the years.
  👉 *Insight:* Northern regions (e.g., Whitehorse) show more extreme fluctuations.

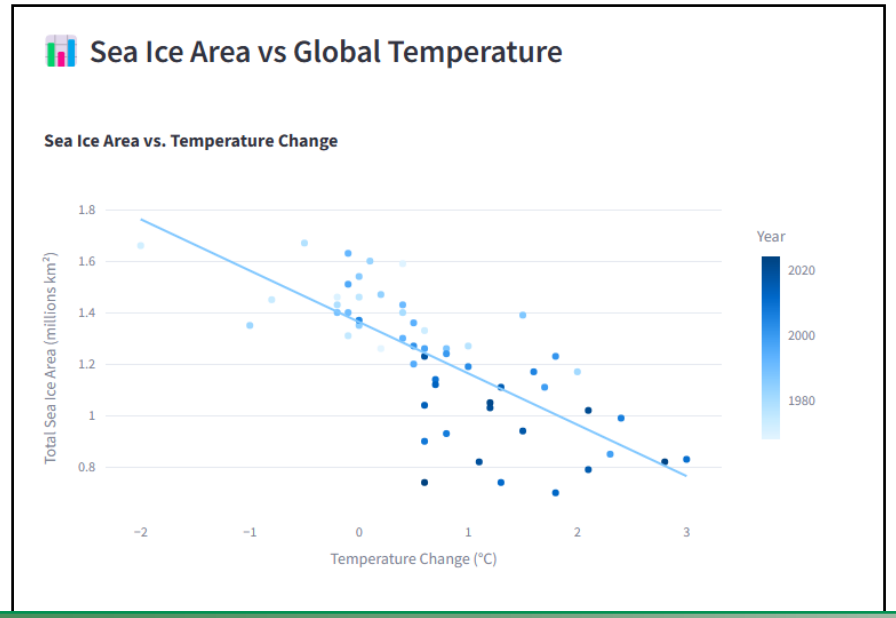


Visualizations - EDA (Exploratory Data Analysis)



Visualizations - EDA (Exploratory Data Analysis)





3. Arctic Sea Ice and Global Temperature Trends

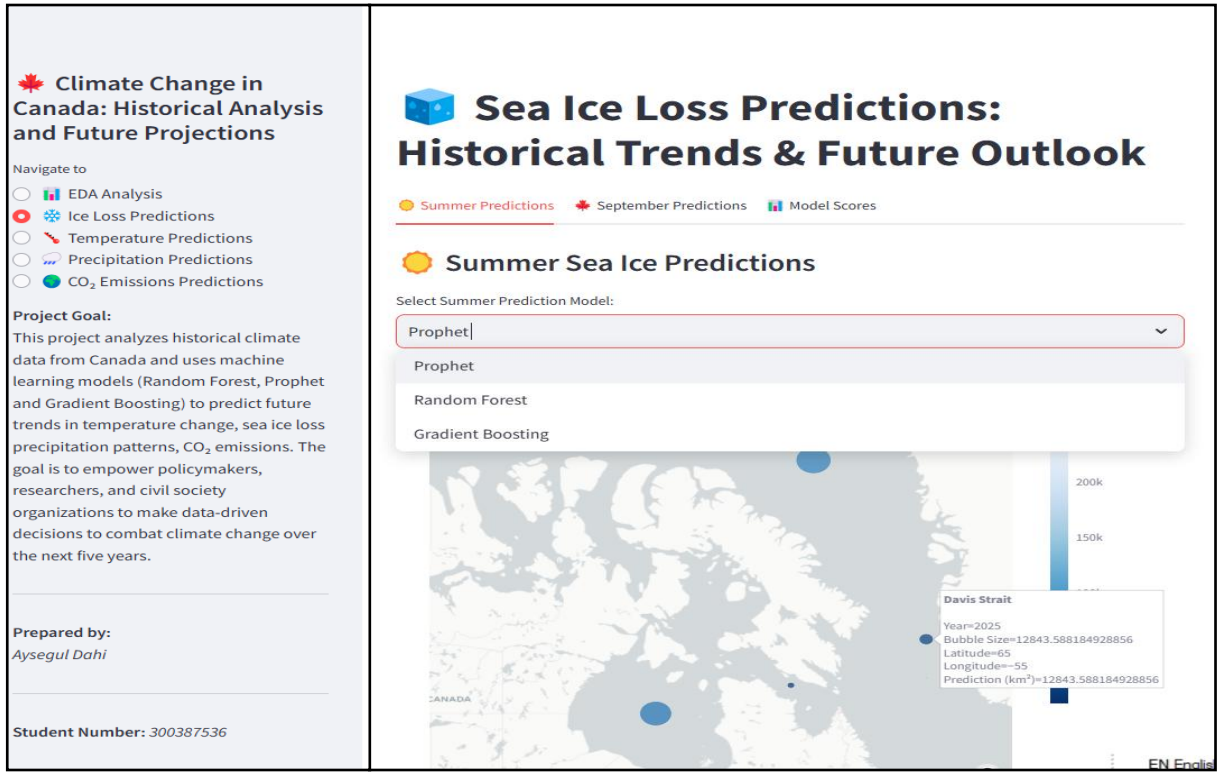
- **Sea Ice Area Line Plot:**
  Visualized the **decline in Arctic sea ice area** from 1968 to 2024.
  👉 *Insight:* Steady and alarming reduction in Arctic sea ice.
- **Global Temperature Change Line Plot:**
  Showed the **rise in global average temperature** across the same period.
  👉 *Insight:* Global temperatures have increased consistently.
- **Scatter Plot: Sea Ice Area vs. Temperature:**
  Demonstrated the **inverse relationship** between **rising temperatures** and **declining sea ice**.
  👉 *Insight:* Clear negative correlation.
- **Correlation Heatmap:**
  Plotted a **heatmap** of correlations between variables (temperature, sea ice area, CO_2 emissions).
  👉 *Insight:* Strong negative correlation between temperature change and sea ice area.

Visualizations – Ice Loss Predictions

In the **Ice Loss Predictions** section of the dashboard, I explore **how Arctic sea ice is expected to change between 2025 and 2030** based on different machine learning models.

I divided the analysis into **Summer** and **September** because these two times represent **critical points** in the Arctic climate system:

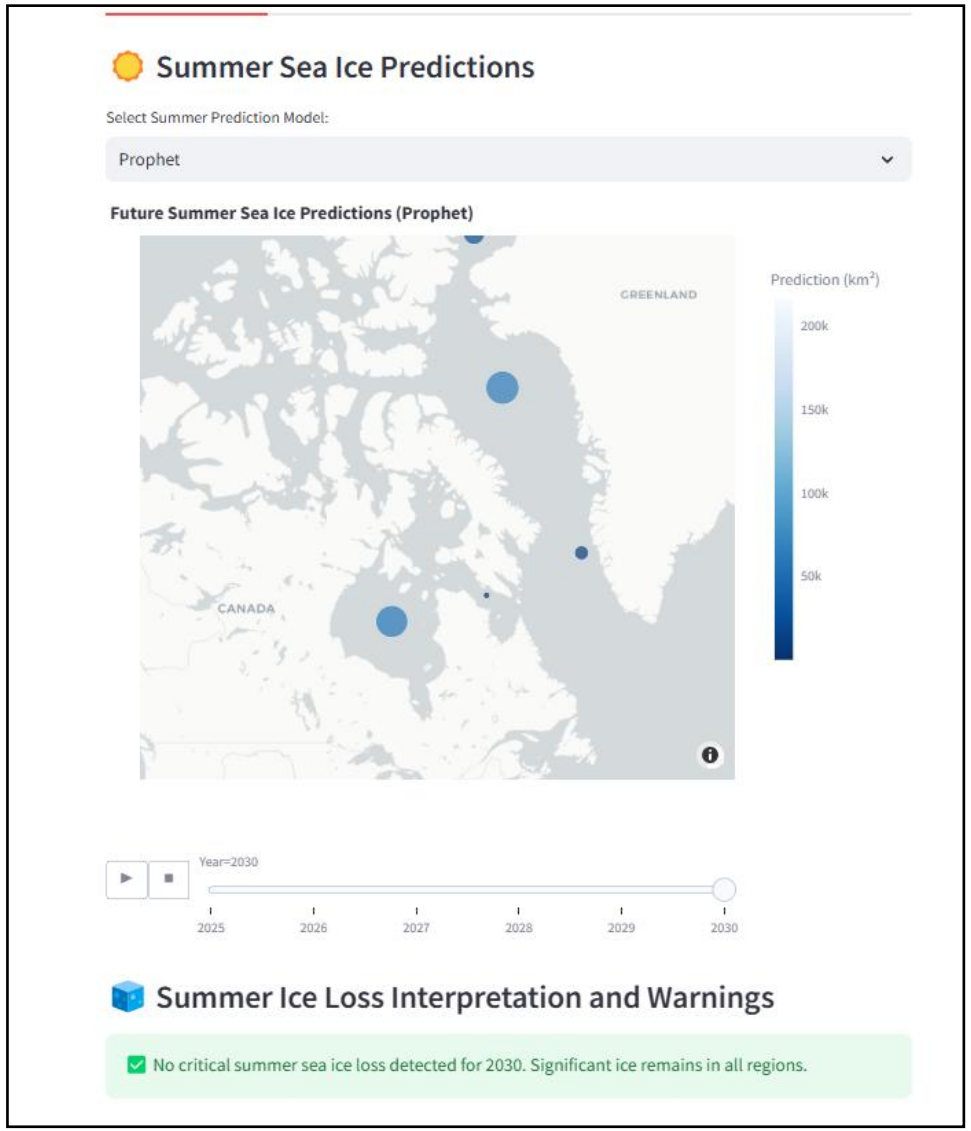
- **Summer (June–August)**: Peak melt season (where sea ice area is at its lowest mid-year).
- **September**: End of melt season minimum (indicator of permanent losses).



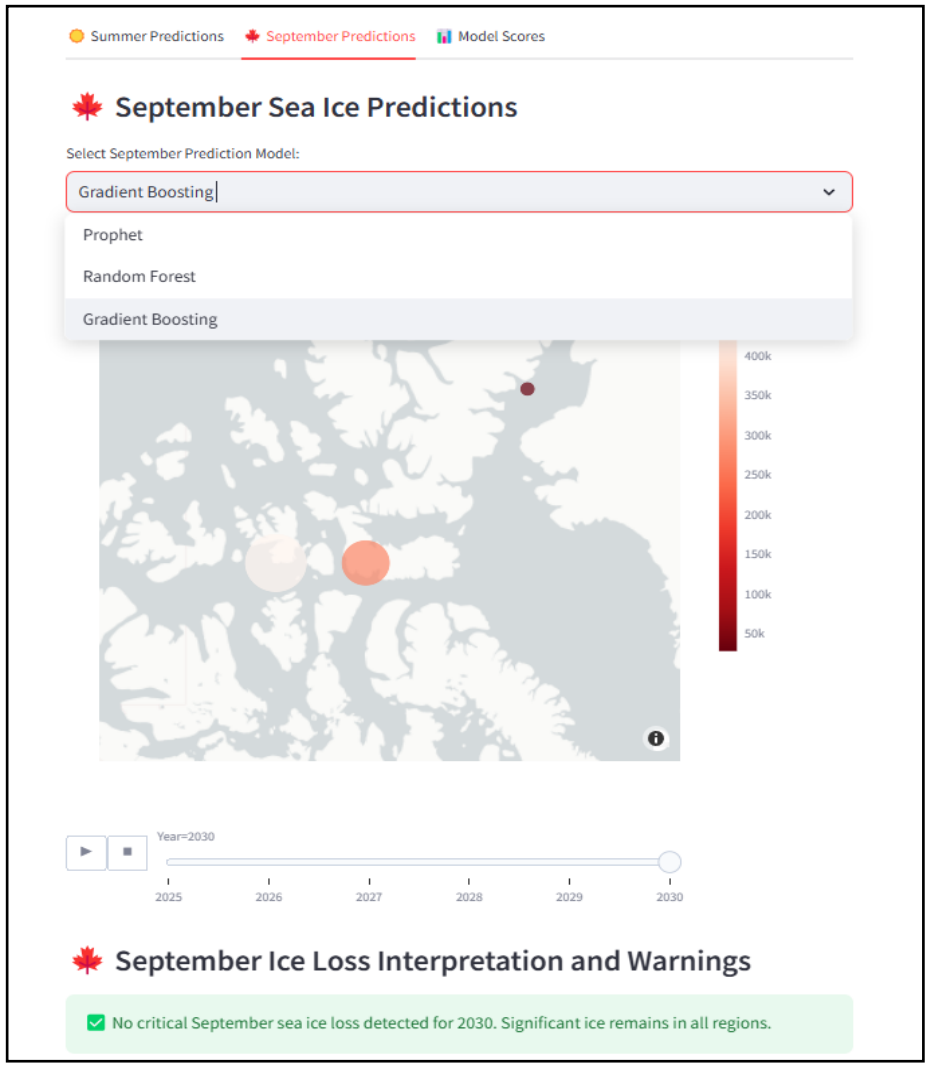
Visualizations – Ice Loss Predictions

1. 🌞 Summer Sea Ice Predictions

- Users can **select different models** (*Random Forest*, *Gradient Boosting*, *Prophet*) to visualize **future summer sea ice predictions**.
- An **animated map** shows **sea ice area predictions** (in thousands of km²) for **different Arctic regions** from 2025 to 2030.
- **Bubble size and color** on the map indicate the amount of remaining sea ice:
  - **Smaller blue bubbles** mean **lower sea ice**.
  - **Larger bubbles** represent **more preserved ice**.
- Critical regions (with **predicted summer ice area below 5,000 km²**) are automatically **highlighted with warnings**.
- Loss of summer sea ice would result in:
  - Collapse of Arctic ecosystems (polar bears, seals).
  - Enhanced global warming due to loss of reflective ice cover (positive feedback loop).
- ✅ *If no region falls below the critical threshold, a success message reassures the user.*



Visualizations – Ice Loss Predictions



2. **September Sea Ice Predictions**

- Similar to the summer predictions, but **focused on September**, which is traditionally the **month with the lowest sea ice extent**.
- **Interactive selection of models** and **animated predictions** on a **red-themed map** show how different regions are projected to change.

Key Interpretation:

- September sea ice is a **strong climate tipping point** indicator.
- If **September ice is lost**, it suggests **permanent Arctic ecosystem changes** and **accelerated global warming**.
- ✅ *Warnings are issued for regions projected to lose almost all September sea ice by 2030.*

Visualizations – Ice Loss Predictions

. 📊 **Model Performance (R^2 Scores)**

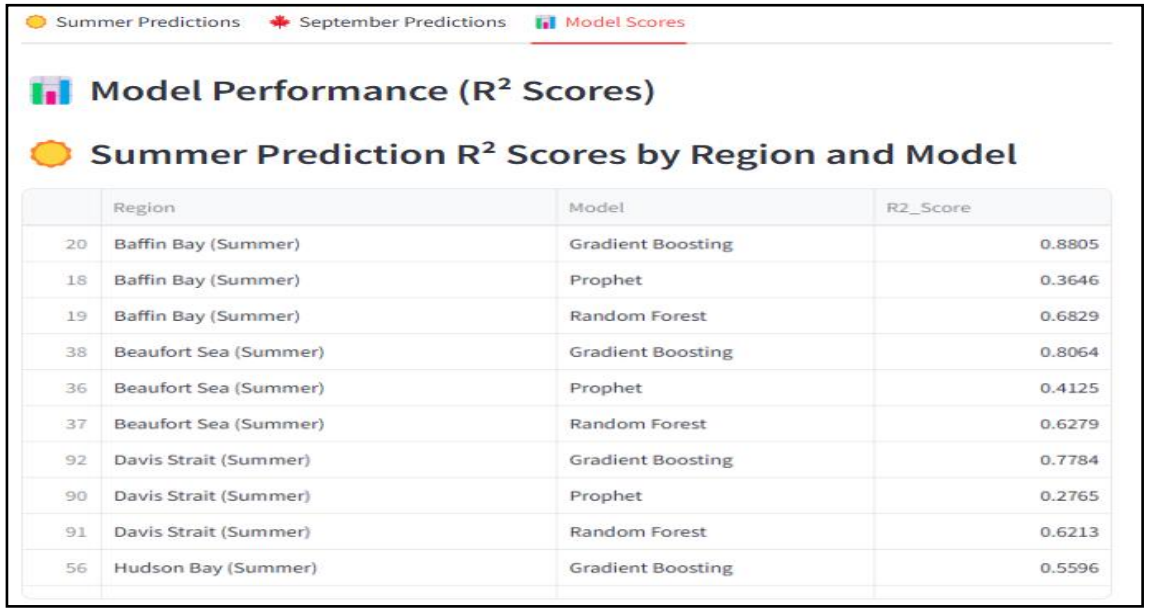
- Two separate tables show **R^2 scores** (goodness of fit) for:
  - **Summer predictions** by region and model.
  - **September predictions** by region and model.

Key Interpretation:

- **Higher R^2 scores** mean better predictive accuracy.
- Some models (typically Gradient Boosting) **outperformed Prophet** and **Random Forest** in specific regions.

Overall Learnings:

- **Clear Arctic sea ice loss trend** is evident by 2030, even under different model assumptions.
- **Certain regions** like **Hudson Bay, Beaufort Sea**, and **Davis Strait** show more vulnerability.
- **Machine Learning models** (Random Forest, Gradient Boosting) provided better regional predictions compared to Prophet in some cases.
- **Visualization of critical loss regions** makes it easy for users to identify where the Arctic system is most at risk.

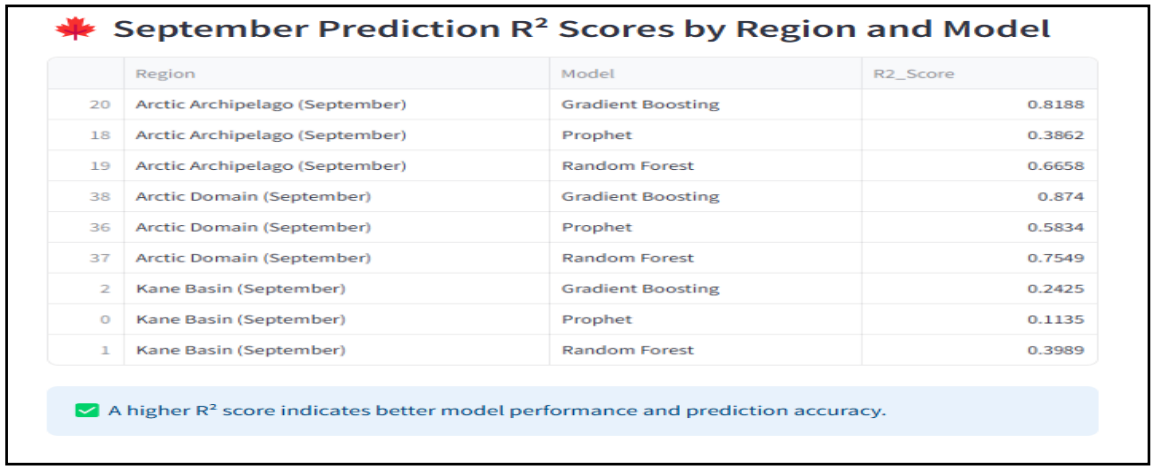


Summer Predictions | September Predictions | Model Scores

Model Performance (R^2 Scores)

Summer Prediction R^2 Scores by Region and Model

	Region	Model	R2_Score
20	Baffin Bay (Summer)	Gradient Boosting	0.8805
18	Baffin Bay (Summer)	Prophet	0.3646
19	Baffin Bay (Summer)	Random Forest	0.6829
38	Beaufort Sea (Summer)	Gradient Boosting	0.8064
36	Beaufort Sea (Summer)	Prophet	0.4125
37	Beaufort Sea (Summer)	Random Forest	0.6279
92	Davis Strait (Summer)	Gradient Boosting	0.7784
90	Davis Strait (Summer)	Prophet	0.2765
91	Davis Strait (Summer)	Random Forest	0.6213
56	Hudson Bay (Summer)	Gradient Boosting	0.5596



September Prediction R^2 Scores by Region and Model

	Region	Model	R2_Score
20	Arctic Archipelago (September)	Gradient Boosting	0.8188
18	Arctic Archipelago (September)	Prophet	0.3862
19	Arctic Archipelago (September)	Random Forest	0.6658
38	Arctic Domain (September)	Gradient Boosting	0.874
36	Arctic Domain (September)	Prophet	0.5834
37	Arctic Domain (September)	Random Forest	0.7549
2	Kane Basin (September)	Gradient Boosting	0.2425
0	Kane Basin (September)	Prophet	0.1135
1	Kane Basin (September)	Random Forest	0.3989

A higher R^2 score indicates better model performance and prediction accuracy.

Visualizations – Ice Loss Predictions

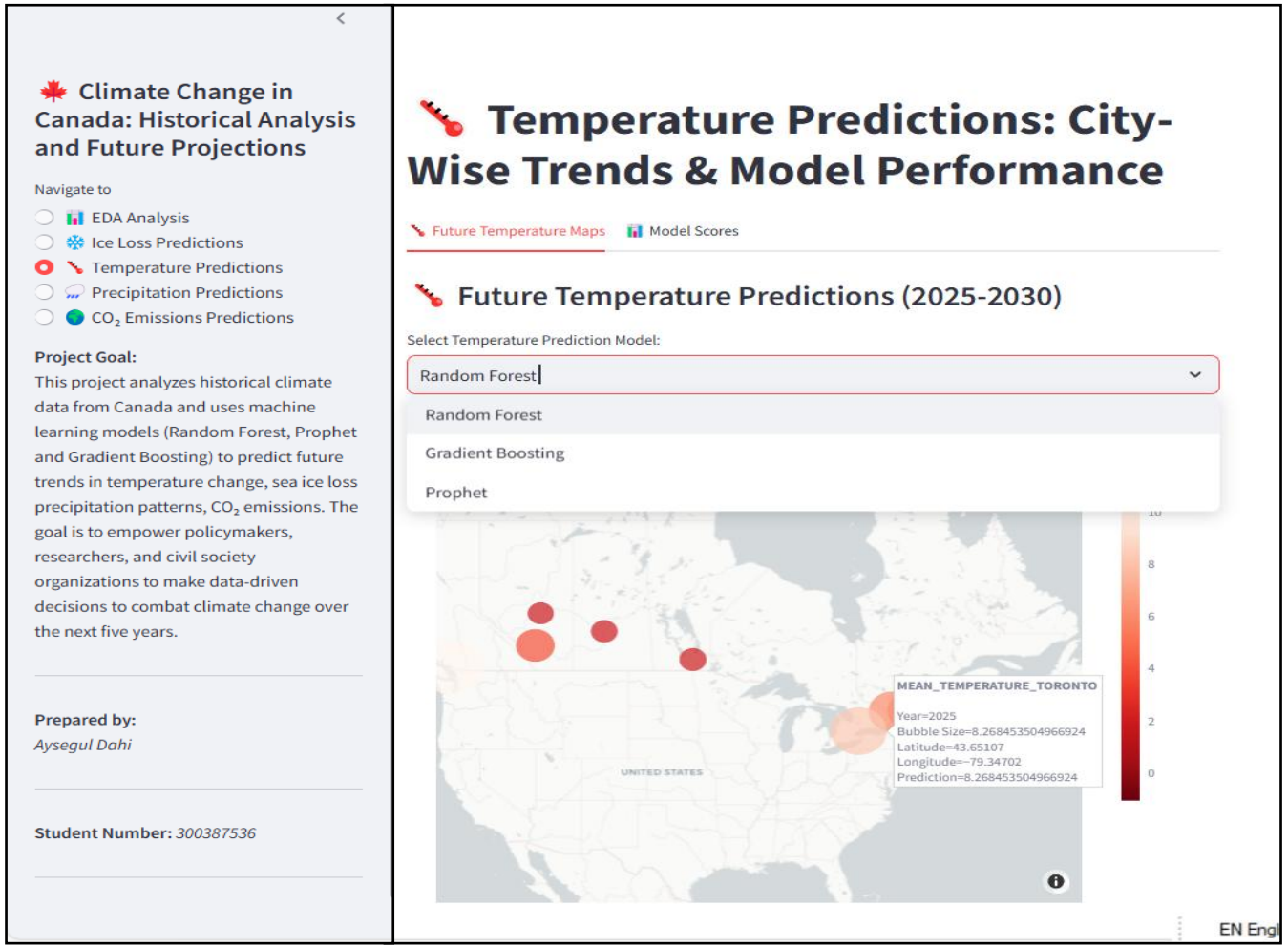
```python
# If Section is "Ice Loss Predictions"
elif section == "❄️ Ice Loss Predictions":
    st.title("🧊 Sea Ice Loss Predictions: Historical Trends & Future Outlook")

    # Load updated datasets
    historical_df = pd.read_csv("csv_files/cleaned_datasets/final_climate_dataset.csv")
    summer_predictions = pd.read_csv("csv_files/predictions_scores/summer_predictions.csv")
    september_predictions = pd.read_csv("csv_files/predictions_scores/september_predictions.csv")

    # Updated Arctic Region Coordinates
    region_coords = {
        "Foxe Basin": (66.5, -78.0),
        "Kane Basin": (79.5, -72.0),
        "Baffin Bay": (74.0, -67.5),
        "Beaufort Sea": (72.0, -140.0),
        "Canadian Arctic Archipelago": (75.0, -90.0),
        "Hudson Bay": (60.0, -85.0),
        "Hudson Strait": (62.0, -70.0),
        "Davis Strait": (65.0, -55.0),
        "Northern Labrador Sea": (62.5, -55.0),
        "Arctic Archipelago": (75.0, -90.0),
        "Arctic Domain": (75.0, -100.0),
    }

    # Tabs
    tab1, tab2, tab3 = st.tabs(["🌞 Summer Predictions", "🍁 September Predictions", "📊 Model Scores"])

    # Summer Predictions Tab
    with tab1:
        st.subheader("🌞 Summer Sea Ice Predictions")

        summer_models = summer_predictions["Model"].unique()
        selected_summer_model = st.selectbox("Select Summer Prediction Model:", summer_models)

        summer_filtered = summer_predictions[summer_predictions["Model"] == selected_summer_model].copy()

        # Clean region names
        summer_filtered["Region Clean"] = summer_filtered["Region"].str.replace(r"\(Summer\)", "", regex=True).str.strip()

        # Map coordinates
        summer_filtered["Latitude"] = summer_filtered["Region Clean"].map(lambda x: region_coords.get(x, (None, None))[0])
        summer_filtered["Longitude"] = summer_filtered["Region Clean"].map(lambda x: region_coords.get(x, (None, None))[1])

        # Convert Predictions to km²
```

```python
    # Summer Map
    fig_summer_pred = px.scatter_mapbox(
        summer_filtered.dropna(subset=["Latitude", "Longitude"]),
        lat="Latitude",
        lon="Longitude",
        size="Bubble Size",
        color="Prediction (km²)",
        hover_name="Region Clean",
        animation_frame="Year",
        color_continuous_scale="Blues_r",
        size_max=40,
        zoom=2.5,
        height=650,
        title=f"Future Summer Sea Ice Predictions ({selected_summer_model})"
    )

    fig_summer_pred.update_layout(mapbox_style="carto-positron", margin={"r": 0, "t": 30, "l": 0, "b": 0})
    st.plotly_chart(fig_summer_pred)

    # Summer Ice Loss Interpretation
    st.subheader("🧊 Summer Ice Loss Interpretation and Warnings")

    critical_loss_summer = summer_filtered[(summer_filtered["Year"] == 2030) & (summer_filtered["Prediction (km²)"] < 5)]

    if not critical_loss_summer.empty:
        st.warning("⚠️ Critical Summer Sea Ice Loss Detected in 2030! Some regions are projected to almost completely lose summer sea ice.")

        for idx, row in critical_loss_summer.iterrows():
            st.write(f"🔹 **Region:** {row['Region Clean']} - **Predicted Summer Sea Ice Area in 2030:** {row['Prediction (km²)']:.2f} km²")

        st.info("""
        **Meaning of Summer Trends:**

        Summer trends represent the minimum sea ice extent during the warmest months.
        A near-complete loss of summer ice means severe ecosystem stress,
        loss of ice-dependent species, and faster warming (positive feedback loops).
        """)
    else:
        st.success("✅ No critical summer sea ice loss detected for 2030. Significant ice remains in all regions.")
September Predictions Tab
ith tab2:
    st.subheader("🍁 September Sea Ice Predictions")

    september_models = september_predictions["Model"].unique()
```

Visualizations – Temperature Change Predictions

In this section, users can explore **future temperature projections** for **Canadian cities** between **2025 and 2030**, based on **three machine learning models**:

- Random Forest
- Gradient Boosting
- Prophet



Visualizations – Temperature Change Predictions

What the User Sees:

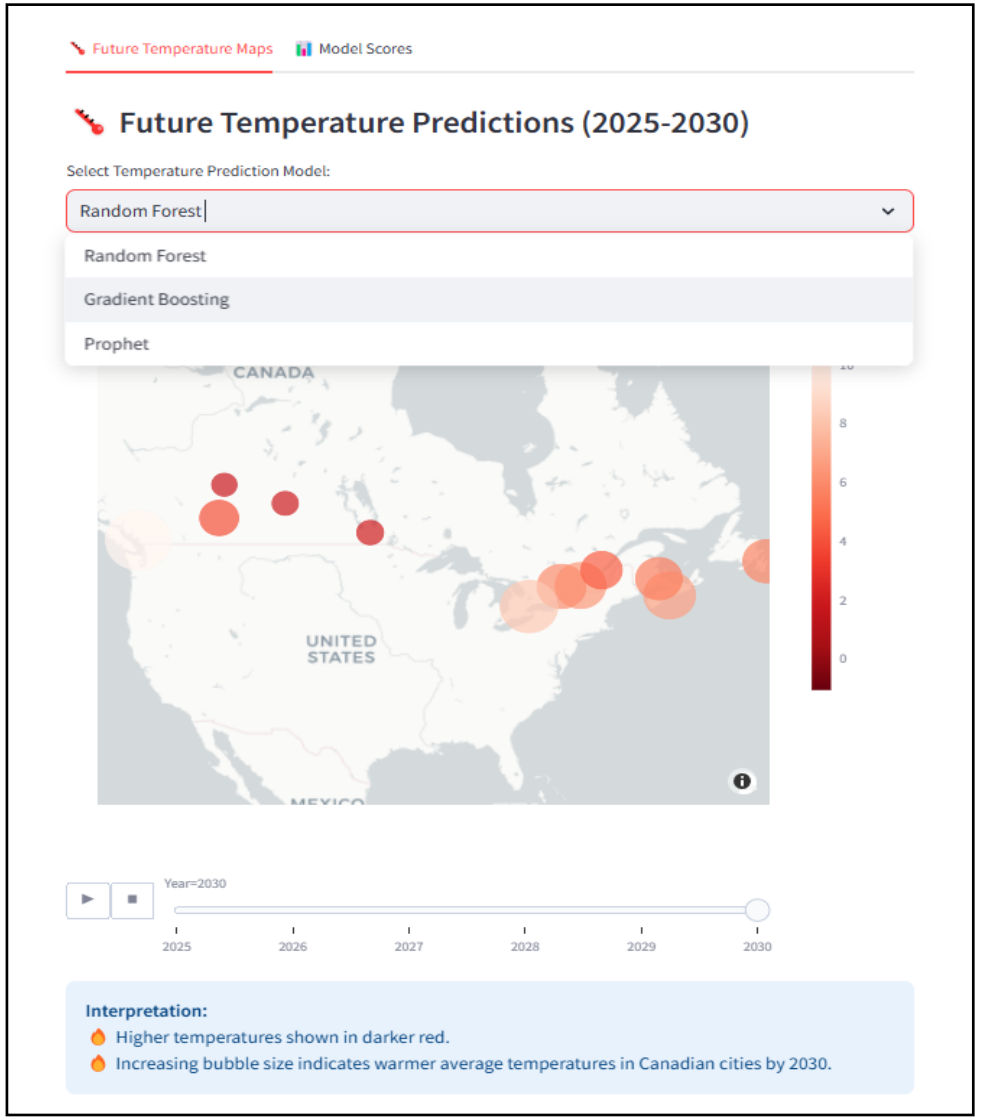
- **Animated Temperature Map:**
  An interactive map shows Canadian cities with bubble sizes representing **predicted temperatures**.

  ➔ **Darker red colors and larger bubbles** indicate **higher predicted average temperatures**.

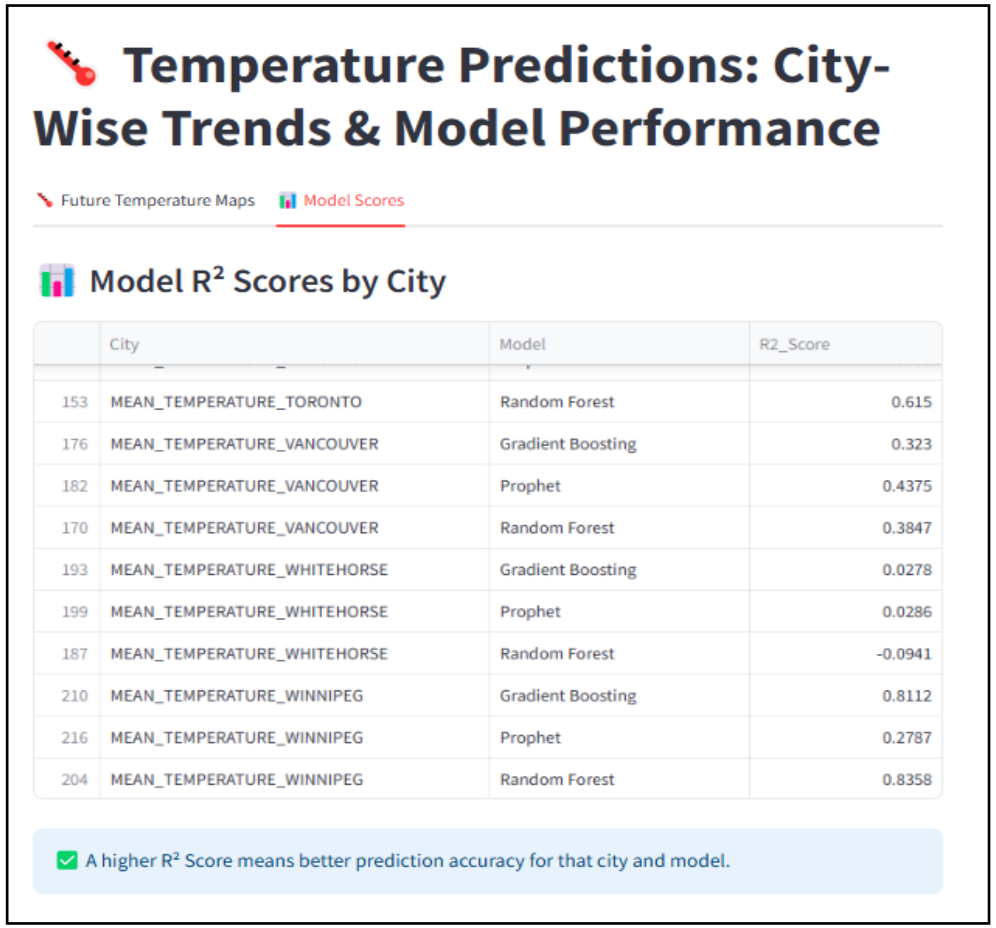
- **Model Selector:**
  Users can select the prediction model (Random Forest, Gradient Boosting, or Prophet) and compare how different models forecast temperature trends.

- **R² Model Scores Table:**
  Users can also view how accurately each model performed historically for each city.

  ➔ **Higher R² Scores** mean better model fit and more reliable predictions.



Visualizations – Temperature Change Predictions



Temperature Predictions: City-Wise Trends & Model Performance

Future Temperature Maps | Model Scores

Model R^2 Scores by City

	City	Model	R2_Score
153	MEAN_TEMPERATURE_TORONTO	Random Forest	0.615
176	MEAN_TEMPERATURE_VANCOUVER	Gradient Boosting	0.323
182	MEAN_TEMPERATURE_VANCOUVER	Prophet	0.4375
170	MEAN_TEMPERATURE_VANCOUVER	Random Forest	0.3847
193	MEAN_TEMPERATURE_WHITEHORSE	Gradient Boosting	0.0278
199	MEAN_TEMPERATURE_WHITEHORSE	Prophet	0.0286
187	MEAN_TEMPERATURE_WHITEHORSE	Random Forest	-0.0941
210	MEAN_TEMPERATURE_WINNIPEG	Gradient Boosting	0.8112
216	MEAN_TEMPERATURE_WINNIPEG	Prophet	0.2787
204	MEAN_TEMPERATURE_WINNIPEG	Random Forest	0.8358

A higher R^2 Score means better prediction accuracy for that city and model.

Interpretation of the Visualizations:

- 🔥 **All Canadian cities show a warming trend toward 2030.**
- 🔥 **Southern cities** like **Toronto**, **Vancouver**, and **Montreal** are expected to experience **moderate to high warming**.
- 🧊 **Northern cities** like **Whitehorse** show **more variability** but still an overall warming trend.
- 📊 In the R^2 Scores table, models like **Gradient Boosting** often show higher R^2 values than Prophet or Random Forest, indicating better predictive performance in most cities.

Key Takeaways:

- Climate warming is consistent across all major Canadian cities.
- Modeling future temperatures helps policymakers plan for urban heat, infrastructure resilience, and environmental strategies.
- Better-performing models (higher R^2) give greater confidence in city-specific forecasts.

Visualizations – Temperature Change Predictions

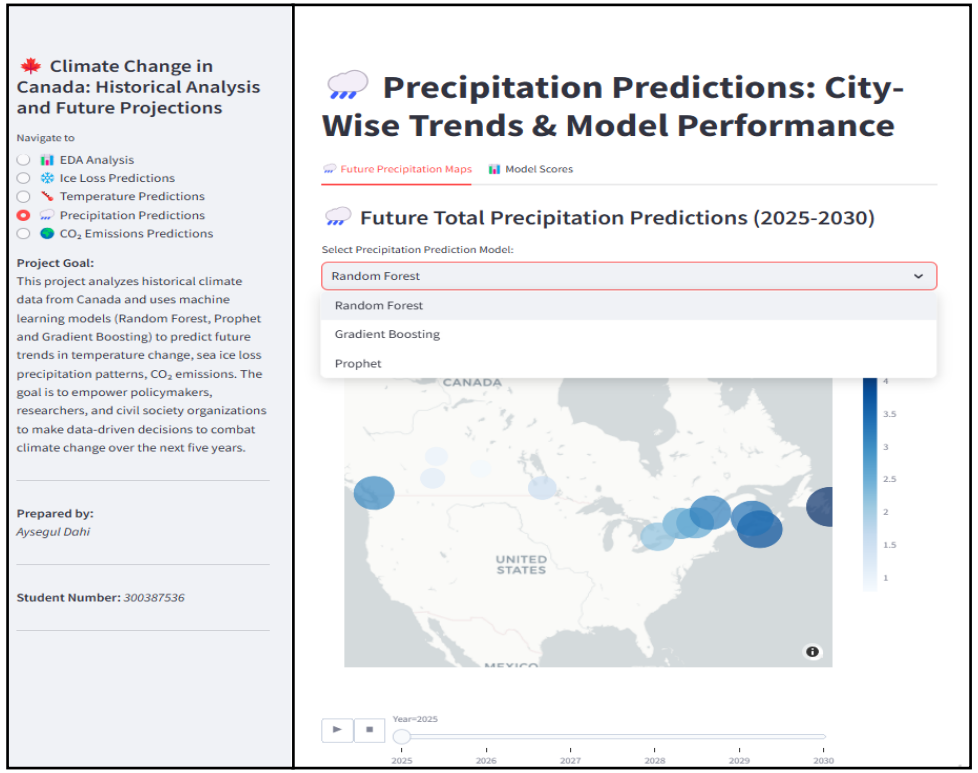
```python
# 🌡️ Temperature Predictions Section

elif section == "🌡️ Temperature Predictions":
    st.title("🌡️ Temperature Predictions: City-Wise Trends & Model Performance")

    # Load data
    temperature_predictions = pd.read_csv('csv_files/predictions_scores/temperature_predictions.csv')

    # City Coordinates (approximate)
    city_coords = {
        "MEAN_TEMPERATURE_CALGARY": (51.0447, -114.0719),
        "MEAN_TEMPERATURE_EDMONTON": (53.5461, -113.4938),
        "MEAN_TEMPERATURE_HALIFAX": (44.6488, -63.5752),
        "MEAN_TEMPERATURE_MONCTON": (46.0878, -64.7782),
        "MEAN_TEMPERATURE_MONTREAL": (45.5017, -73.5673),
        "MEAN_TEMPERATURE_OTTAWA": (45.4215, -75.6972),
        "MEAN_TEMPERATURE_QUEBEC": (46.8139, -71.2082),
        "MEAN_TEMPERATURE_SASKATOON": (52.1579, -106.6702),
        "MEAN_TEMPERATURE_STJOHNS": (47.5615, -52.7126),
        "MEAN_TEMPERATURE_TORONTO": (43.65107, -79.347015),
        "MEAN_TEMPERATURE_VANCOUVER": (49.2827, -123.1207),
        "MEAN_TEMPERATURE_WHITEHORSE": (60.7212, -135.0568),
        "MEAN_TEMPERATURE_WINNIPEG": (49.8951, -97.1384)
    }

    # Tabs
    tab1, tab2 = st.tabs(["🌡️ Future Temperature Maps", "📊 Model Scores"])

    # Future Temperature Maps
    with tab1:
        st.subheader("🌡️ Future Temperature Predictions (2025-2030)")

        # Model Selector
        temp_models = temperature_predictions["Model"].unique()
        selected_temp_model = st.selectbox("Select Temperature Prediction Model:", temp_models)

        temp_filtered = temperature_predictions[temperature_predictions["Model"] == selected_temp_model].copy()

        # Clean City Names
        temp_filtered["Latitude"] = temp_filtered["City"].map(lambda x: city_coords.get(x, (None, None))[0])
        temp_filtered["Longitude"] = temp_filtered["City"].map(lambda x: city_coords.get(x, (None, None))[1])

        # Bubble Size (clip negatives)
        temp_filtered["Bubble Size"] = temp_filtered["Prediction"].clip(lower=0)
```

```python
        # Bubble Size (clip negatives)
        temp_filtered["Bubble Size"] = temp_filtered["Prediction"].clip(lower=0)

        # Map
        fig_temp_pred = px.scatter_mapbox(
            temp_filtered.dropna(subset=["Latitude", "Longitude"]),
            lat="Latitude",
            lon="Longitude",
            size="Bubble Size",
            color="Prediction",
            hover_name="City",
            animation_frame="Year",
            color_continuous_scale="Reds_r",
            size_max=40,
            zoom=2.5,
            height=650,
            title="Predicted Future Temperatures (2025-2030)"
        )

        fig_temp_pred.update_layout(mapbox_style="carto-positron", margin={"r":0,"t":30,"l":0,"b":0})
        st.plotly_chart(fig_temp_pred)

        # Optional interpretation if you want:
        st.info("""
        **Interpretation:**
        🔥 Higher temperatures shown in darker red.
        🔥 Increasing bubble size indicates warmer average temperatures in Canadian cities by 2030.
        """)

    # 📊 Model Scores
    with tab2:
        st.subheader("📊 Model R² Scores by City")

        temp_scores = temperature_predictions[["City", "Model", "R2_Score"]].drop_duplicates()
        st.dataframe(temp_scores.sort_values(["City", "Model"]))

        st.info("✅ A higher R² Score means better prediction accuracy for that city and model.")
# Precipitation Predictions Section

elif section == "🌧️ Precipitation Predictions":
    st.title("🌧️ Precipitation Predictions: City-Wise Trends & Model Performance")

    # Load data
    precipitation_predictions = pd.read_csv('csv_files/predictions_scores/precipitation_predictions.csv')

    # City Coordinates (approximate)
    precipitation_coords = {
        "TOTAL_PRECIPITATION_CALGARY": (51.0447, -114.0719),
        "TOTAL_PRECIPITATION_EDMONTON": (53.5461, -113.4938),
        "TOTAL_PRECIPITATION_HALIFAX": (44.6488, -63.5752),
        "TOTAL_PRECIPITATION_MONCTON": (46.0878, -64.7782),
        "TOTAL_PRECIPITATION_MONTREAL": (45.5017, -73.5673),
        "TOTAL_PRECIPITATION_OTTAWA": (45.4215, -75.6972),
        "TOTAL_PRECIPITATION_QUEBEC": (46.8139, -71.2082),
        "TOTAL_PRECIPITATION_SASKATOON": (52.1579, -106.6702),
```

Visualizations – Precipitation Predictions

In this section, users can **explore the future of precipitation patterns** across Canadian cities from **2025 to 2030**.
Three machine learning models (Random Forest, Gradient Boosting, and Prophet) have been trained and evaluated to make these forecasts. Users can:

- Visualize **how precipitation might change** geographically using an **animated map**.
- Compare **model performances** (via R^2 scores) city-by-city.



Visualizations – Precipitation Predictions

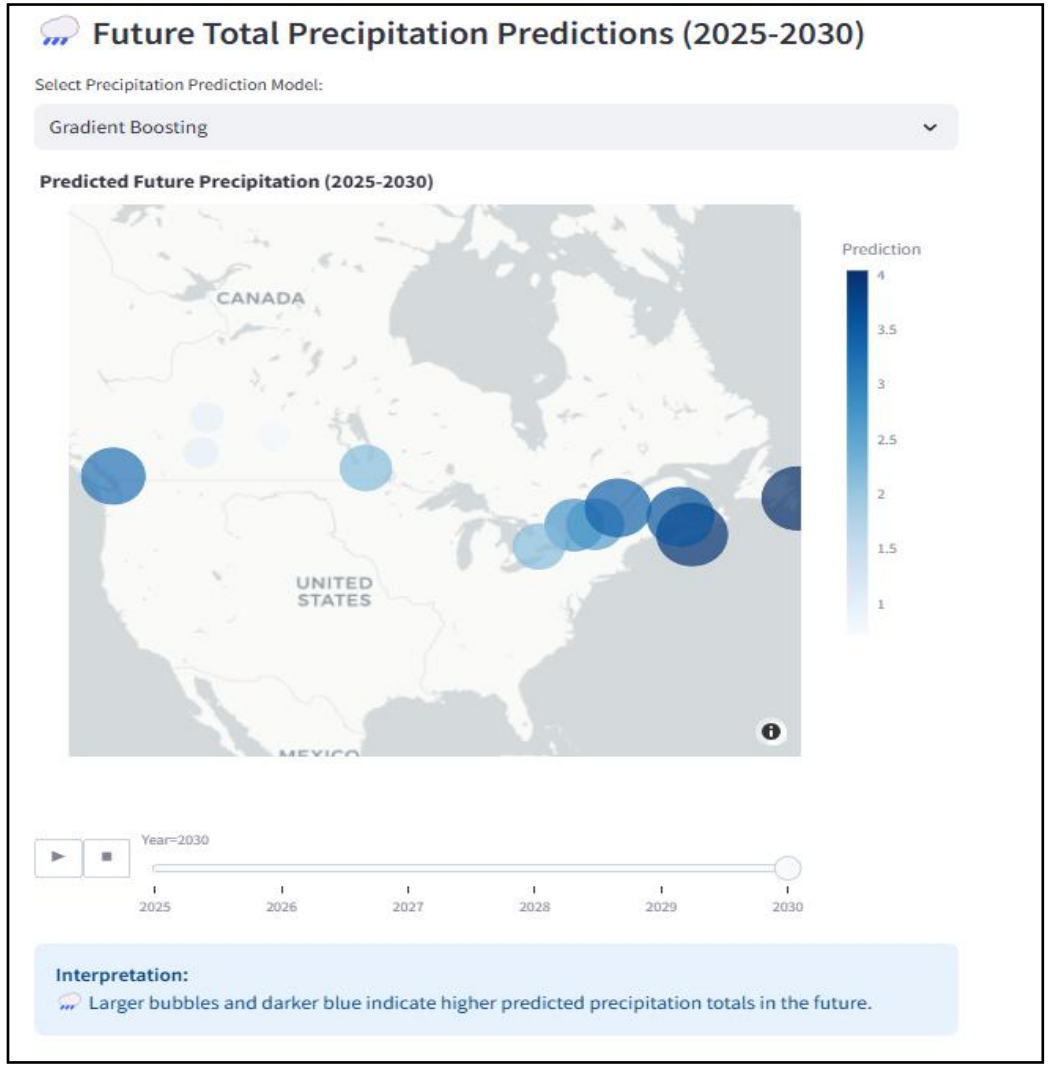
What Users Will See

Future Precipitation Maps (2025–2030)

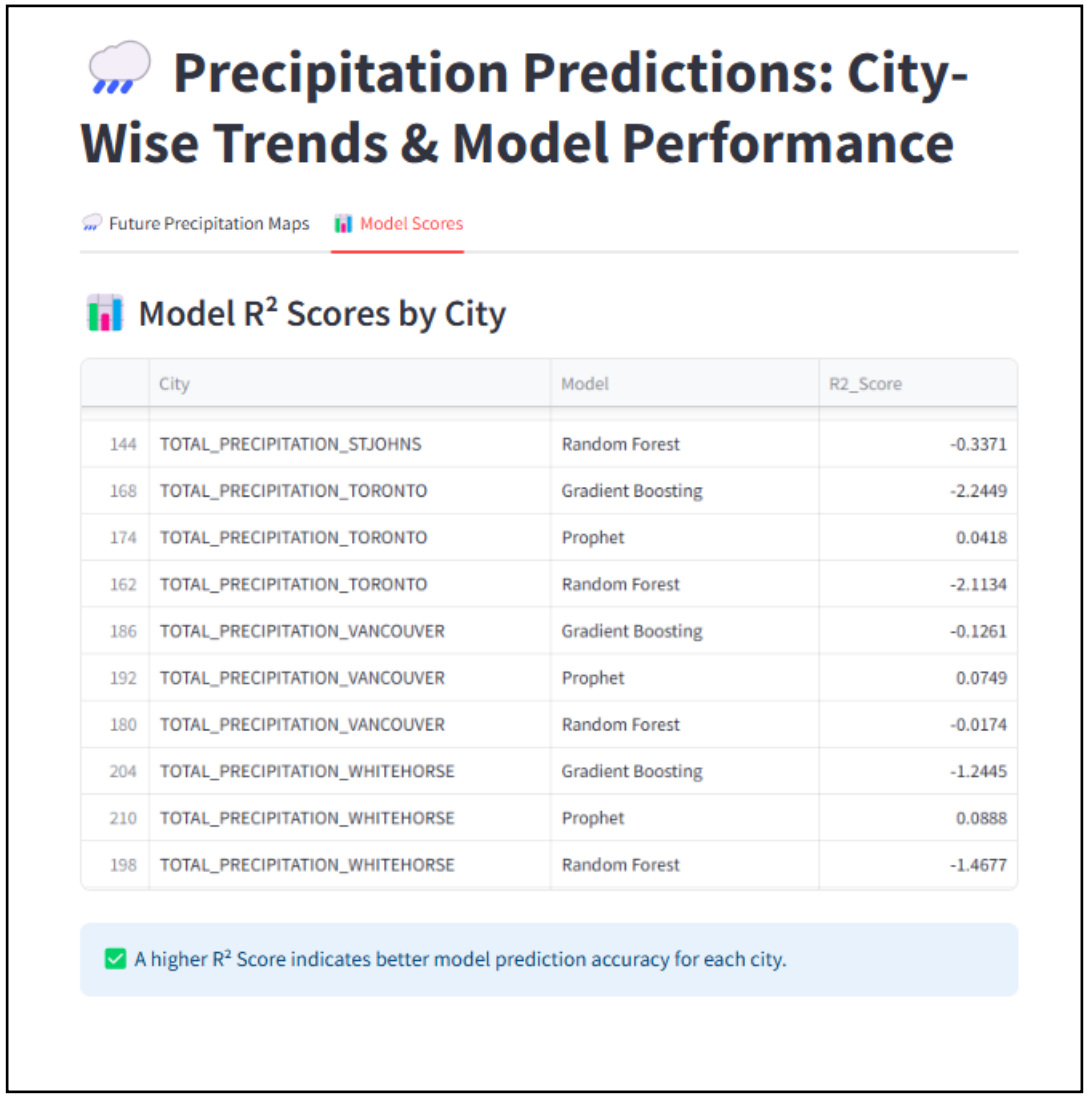
- An **interactive animated map** showing predicted total precipitation levels across Canadian cities.
- Each city is represented by a **bubble** where:
  - **Bubble Size**: Higher predicted precipitation = bigger bubble.
  - **Bubble Color**: Darker blue = more precipitation.
- Users can **select a machine learning model** to see its specific predictions (Random Forest, Gradient Boosting, or Prophet).

Model Scores Tab

- A **table listing R^2 Scores** for each model and city.
- This helps users **judge model performance**:
  - **Higher R^2 Score** = Better prediction reliability.



Visualizations – Precipitation Predictions



Precipitation Predictions: City-Wise Trends & Model Performance

Future Precipitation Maps | Model Scores

Model R^2 Scores by City

	City	Model	R2_Score
144	TOTAL_PRECIPITATION_STJOHNS	Random Forest	-0.3371
168	TOTAL_PRECIPITATION_TORONTO	Gradient Boosting	-2.2449
174	TOTAL_PRECIPITATION_TORONTO	Prophet	0.0418
162	TOTAL_PRECIPITATION_TORONTO	Random Forest	-2.1134
186	TOTAL_PRECIPITATION_VANCOUVER	Gradient Boosting	-0.1261
192	TOTAL_PRECIPITATION_VANCOUVER	Prophet	0.0749
180	TOTAL_PRECIPITATION_VANCOUVER	Random Forest	-0.0174
204	TOTAL_PRECIPITATION_WHITEHORSE	Gradient Boosting	-1.2445
210	TOTAL_PRECIPITATION_WHITEHORSE	Prophet	0.0888
198	TOTAL_PRECIPITATION_WHITEHORSE	Random Forest	-1.4677

A higher R^2 Score indicates better model prediction accuracy for each city.

Key Insights

- **Regional Variations**:
  Some cities are predicted to experience significant increases in precipitation, while others remain relatively stable.
- **Model Performance**:
  Precipitation models generally have **lower R^2 scores** compared to temperature models, indicating **higher uncertainty** in forecasting precipitation.
- **Practical Importance**:
  Understanding future precipitation patterns is critical for:
  - **Flood risk management**
  - **Urban planning**
  - **Agricultural planning**
  - **Climate adaptation policies**

Visualizations – Precipitation Predictions

```python
elif section == "🌧️ Precipitation Predictions":
    st.title("🌧️ Precipitation Predictions: City-Wise Trends & Model Performance")

    # Load data
    precipitation_predictions = pd.read_csv('csv_files/predictions_scores/precipitation_predictions.csv')

    # City Coordinates (approximate)
    precipitation_coords = {
        "TOTAL_PRECIPITATION_CALGARY": (51.0447, -114.0719),
        "TOTAL_PRECIPITATION_EDMONTON": (53.5461, -113.4938),
        "TOTAL_PRECIPITATION_HALIFAX": (44.6488, -63.5752),
        "TOTAL_PRECIPITATION_MONCTON": (46.0878, -64.7782),
        "TOTAL_PRECIPITATION_MONTREAL": (45.5017, -73.5673),
        "TOTAL_PRECIPITATION_OTTAWA": (45.4215, -75.6972),
        "TOTAL_PRECIPITATION_QUEBEC": (46.8139, -71.2082),
        "TOTAL_PRECIPITATION_SASKATOON": (52.1579, -106.6702),
        "TOTAL_PRECIPITATION_STJOHNS": (47.5615, -52.7126),
        "TOTAL_PRECIPITATION_TORONTO": (43.65107, -79.347015),
        "TOTAL_PRECIPITATION_VANCOUVER": (49.2827, -123.1207),
        "TOTAL_PRECIPITATION_WHITEHORSE": (60.7212, -135.0568),
        "TOTAL_PRECIPITATION_WINNIPEG": (49.8951, -97.1384)
    }

    # Tabs
    tab1, tab2 = st.tabs(["🌧️ Future Precipitation Maps", "📊 Model Scores"])
    # Future Precipitation Maps
    with tab1:
        st.subheader("🌧️ Future Total Precipitation Predictions (2025-2030)")

        # Model Selector
        precipitation_models = precipitation_predictions["Model"].unique()
        selected_precipitation_model = st.selectbox("Select Precipitation Prediction Model:", precipitation_models)

        precipitation_filtered = precipitation_predictions[precipitation_predictions["Model"] == selected_precipitation_model].copy()
```

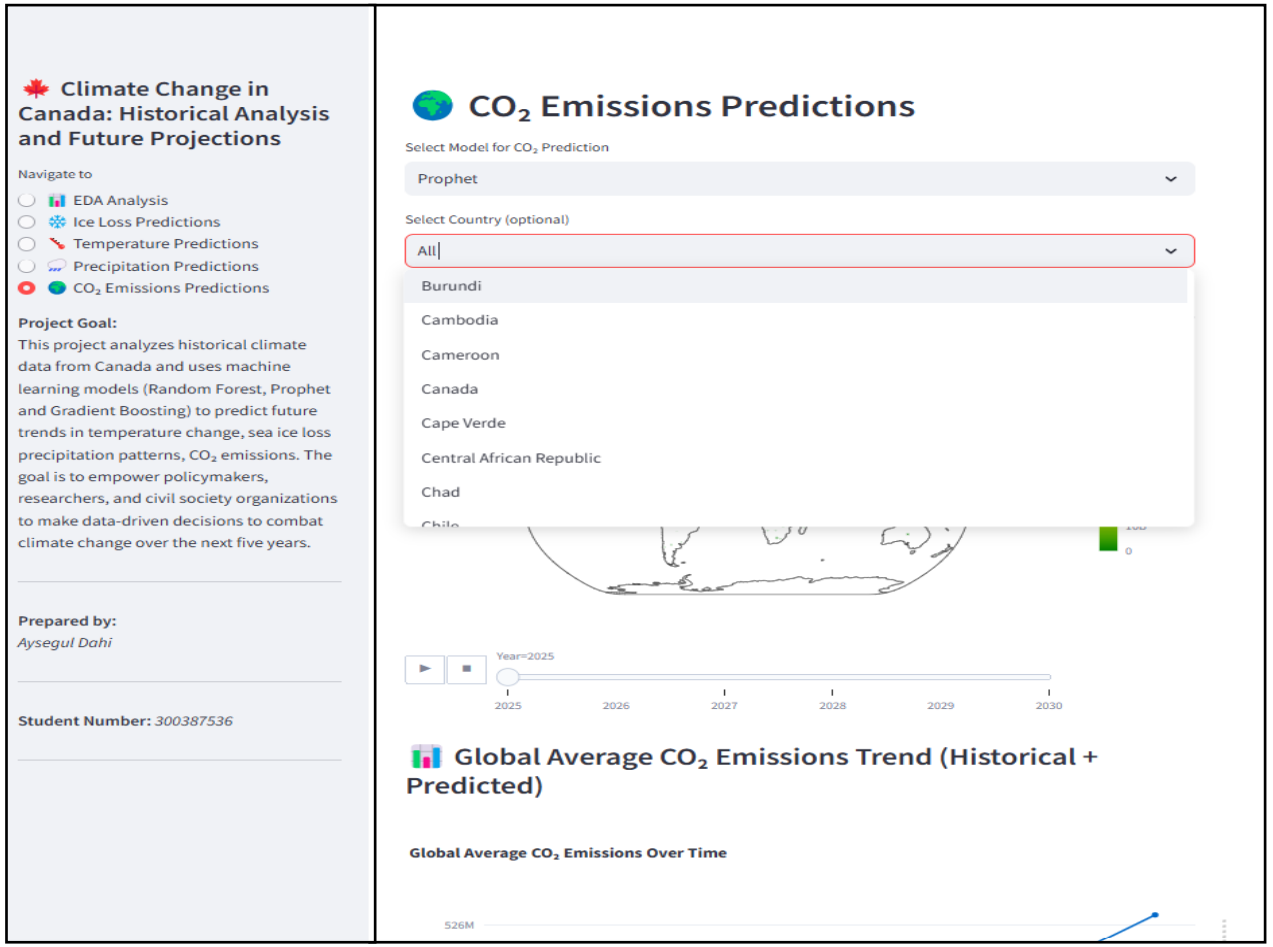
```python
        # Clean City Names
        precipitation_filtered["Latitude"] = precipitation_filtered["City"].map(lambda x: precipitation_coords.get(x, (None, None))[0])
        precipitation_filtered["Longitude"] = precipitation_filtered["City"].map(lambda x: precipitation_coords.get(x, (None, None))[1])

        # Bubble Size (clip negatives)
        precipitation_filtered["Bubble Size"] = precipitation_filtered["Prediction"].clip(lower=0)

        # Map
        fig_precipitation_pred = px.scatter_mapbox(
            precipitation_filtered.dropna(subset=["Latitude", "Longitude"]),
            lat="Latitude",
            lon="Longitude",
            size="Bubble Size",
            color="Prediction",
            hover_name="City",
            animation_frame="Year",
            color_continuous_scale="Blues",
            size_max=40,
            zoom=2.5,
            height=650,
            title="Predicted Future Precipitation (2025-2030)"
        )

        fig_precipitation_pred.update_layout(mapbox_style="carto-positron", margin={"r":0,"t":30,"l":0,"b":0})
        st.plotly_chart(fig_precipitation_pred)

        st.info("""
        **Interpretation:**
        🌧️ Larger bubbles and darker blue indicate higher predicted precipitation totals in the future.
        """)

    # Model Scores
    with tab2:
        st.subheader("📊 Model R² Scores by City")

        precipitation_scores = precipitation_predictions[["City", "Model", "R2_Score"]].drop_duplicates()
        st.dataframe(precipitation_scores.sort_values(["City", "Model"]))

        st.info("✅ A higher R² Score indicates better model prediction accuracy for each city.")

    # Model Scores
```

Visualizations – CO2 Predictions

This section allows users to explore global CO_2 emissions forecasts between 2025 and 2030 based on historical trends and machine learning predictions.

Using this page, users can:

- Visualize future global CO_2 emissions geographically.
- Zoom into specific countries to see their individual predictions.
- Analyze model performance (R^2 scores) across different countries.
- Understand the global trend of CO_2 emissions over time.



Visualizations – CO2 Predictions

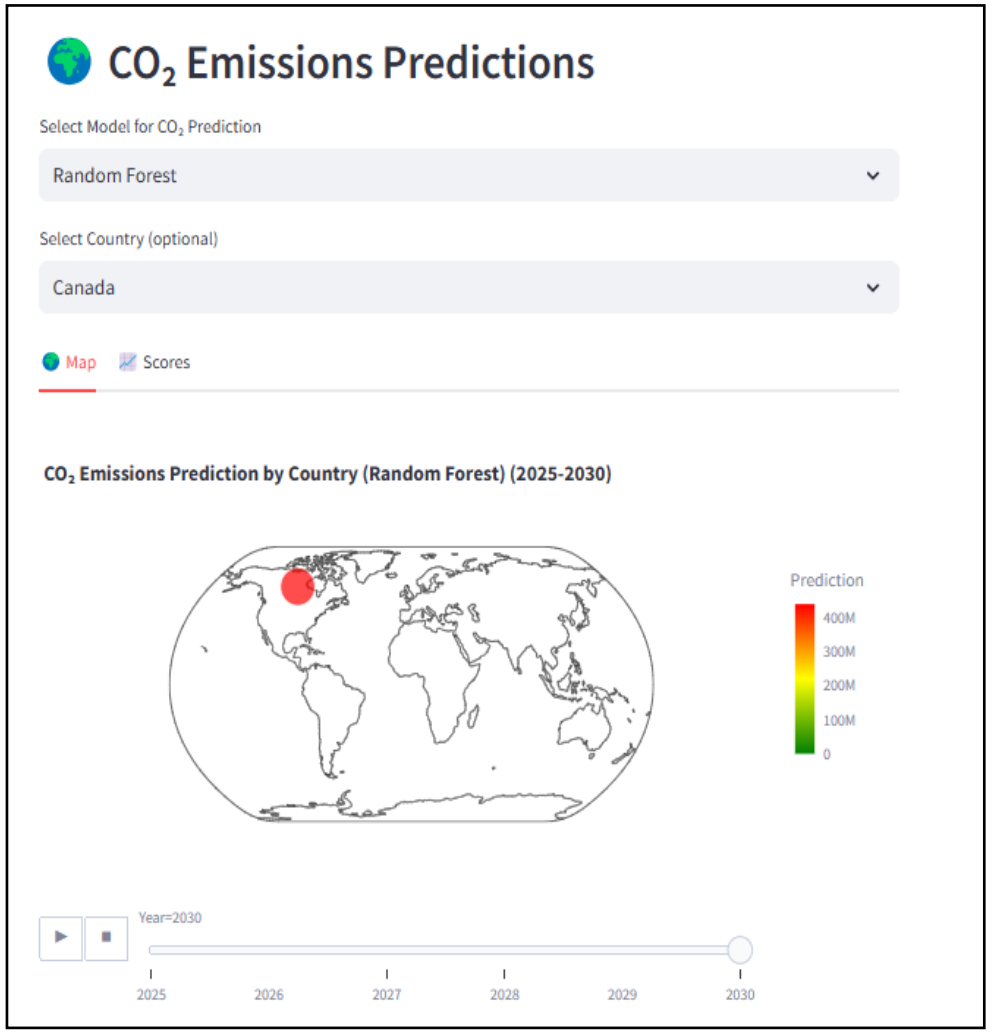
What Users Will See

Global CO_2 Emissions Map (2025–2030)

- **Animated world map** showing **predicted CO_2 emissions** by country.
- **Bubble Size**: Larger bubbles = higher predicted emissions.
- **Color Gradient**:
  - **Green** = Lower emissions
  - **Yellow** = Moderate emissions
  - **Red** = High emissions
- Users can:
  - **Select a machine learning model**
  - (Prophet, Random Forest, Gradient Boosting).
  - **Focus on a specific country** or view the entire world.

Model Scores Tab

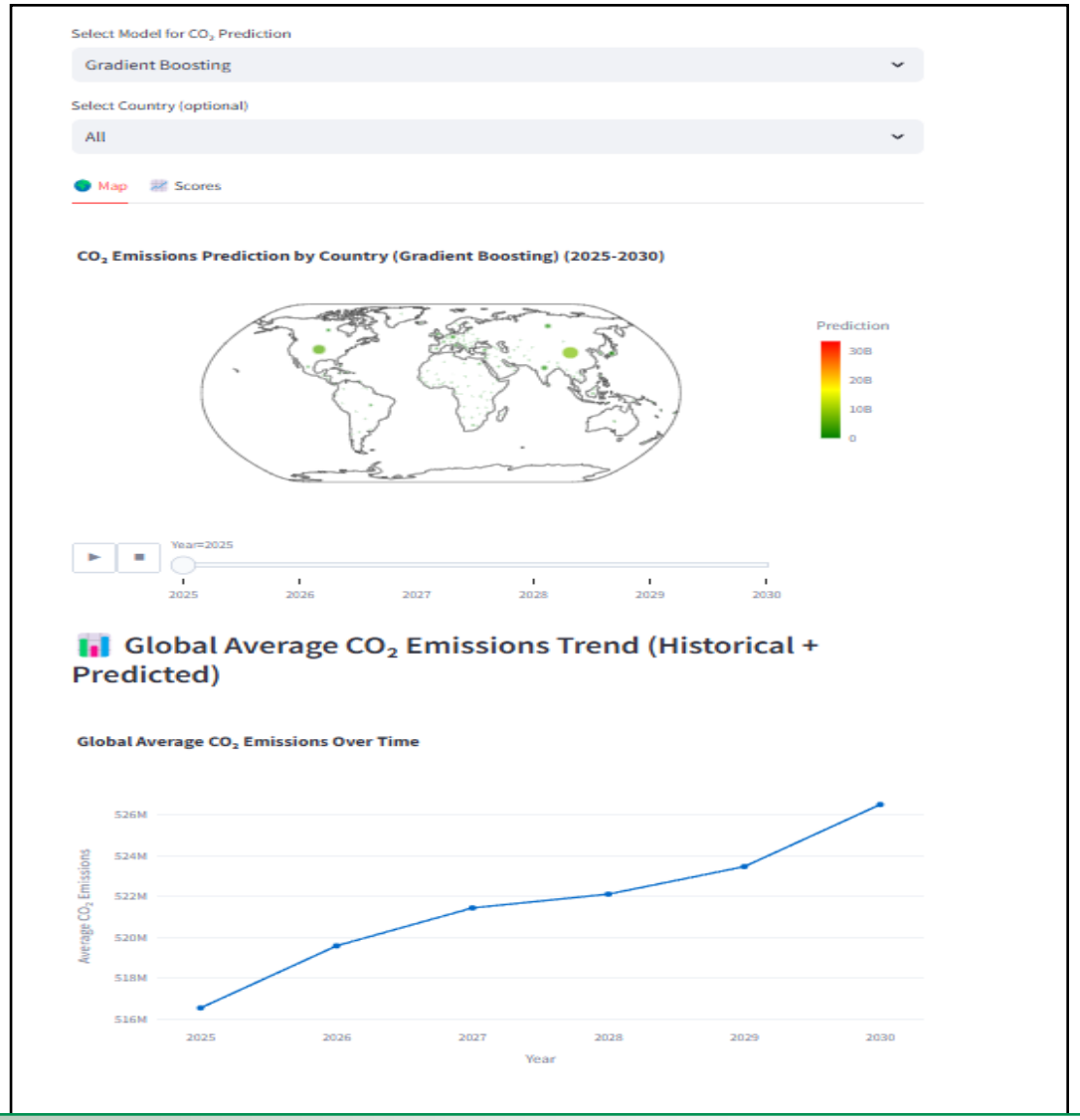
- A **table of R^2 scores** by country and model.
- Higher R^2 scores indicate better predictive performance.



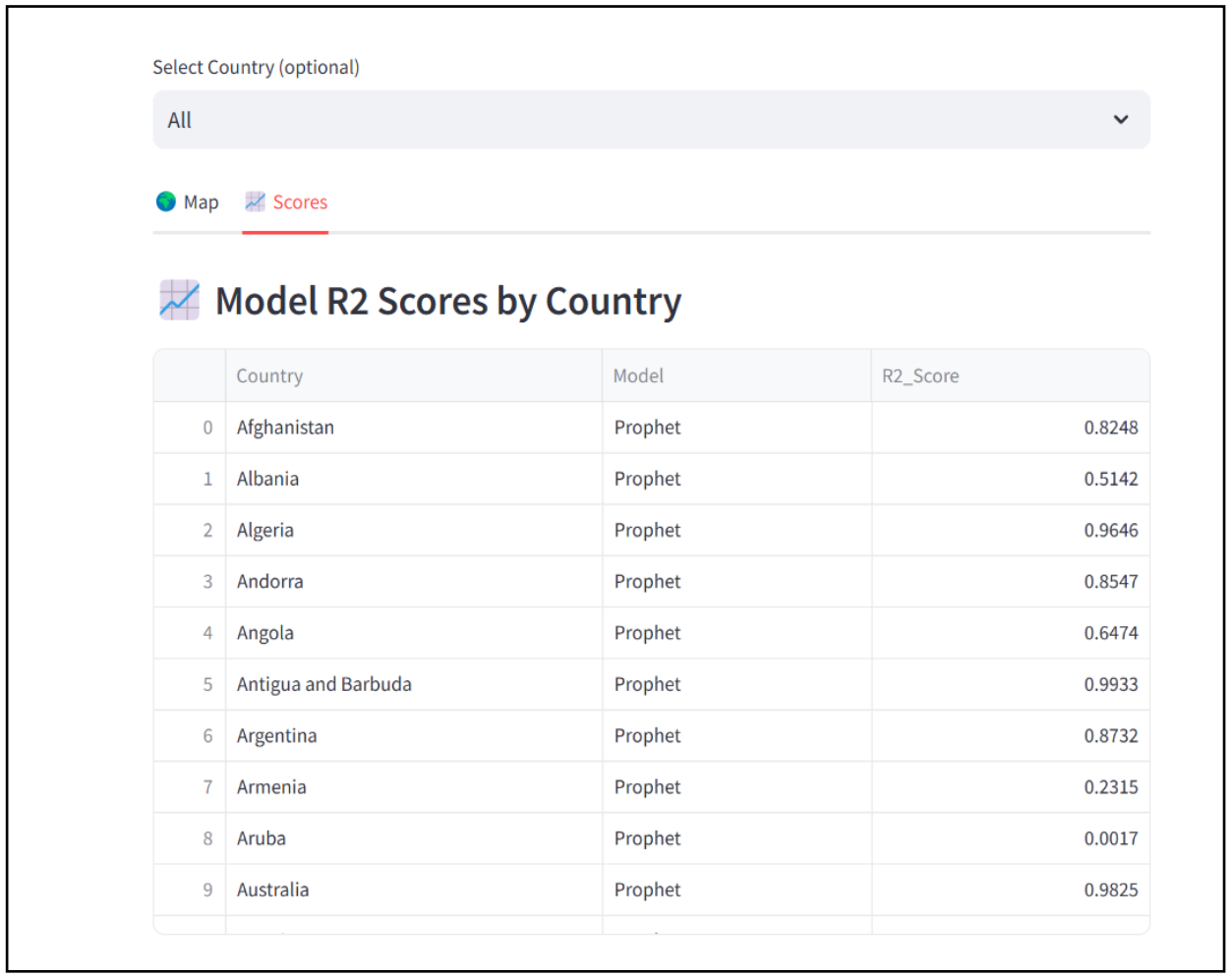
Visualizations – CO2 Predictions

Global Trend Line

- A line chart displaying the global average of predicted CO_2 emissions from historical years through 2030.
- Helps visualize whether global emissions are increasing or decreasing.



Visualizations – CO2 Predictions



	Country	Model	R2_Score
0	Afghanistan	Prophet	0.8248
1	Albania	Prophet	0.5142
2	Algeria	Prophet	0.9646
3	Andorra	Prophet	0.8547
4	Angola	Prophet	0.6474
5	Antigua and Barbuda	Prophet	0.9933
6	Argentina	Prophet	0.8732
7	Armenia	Prophet	0.2315
8	Aruba	Prophet	0.0017
9	Australia	Prophet	0.9825

Key Insights

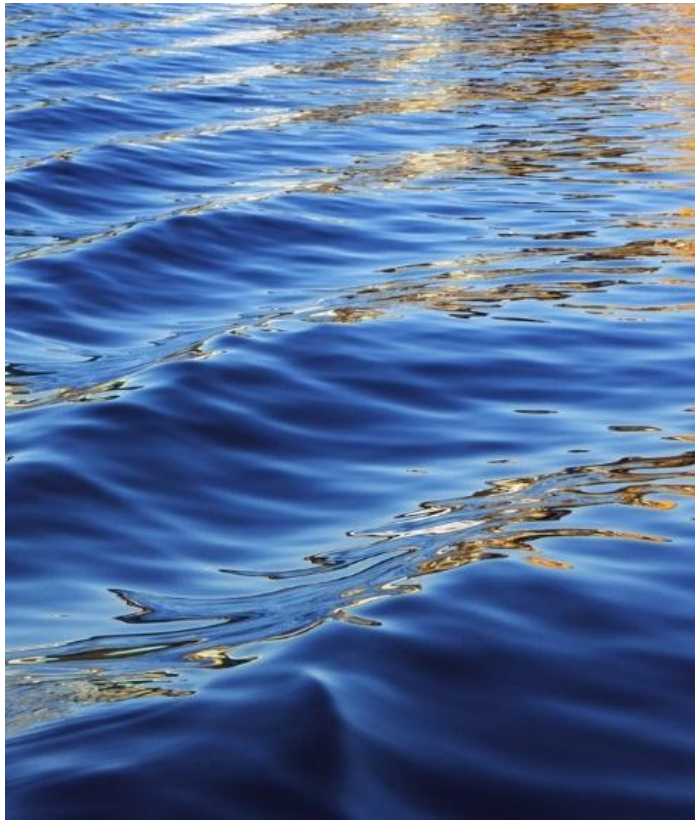
- **Regional Variations**:
  Some regions, especially **industrialized countries**, continue to have **high emissions projections** despite climate initiatives.
- **Model Performance**:
  In most countries, **Random Forest** and **Gradient Boosting** models achieved **very high R^2 scores** (>0.99), suggesting strong prediction reliability.
- **Practical Importance**:
  Understanding future CO_2 emissions is essential for:
  - **International climate agreements**
  - **National policy making**
  - **Setting and meeting emissions reduction targets**

Visualizations – CO2 Predictions

```python
#CO₂ Emissions Predictions Section
elif section == "🌎 CO₂ Emissions Predictions":
    st.header("🌎 CO₂ Emissions Predictions")

    # Load CO2 emissions data
    co2_data = pd.read_csv('csv_files/cleaned_datasets/co2_cleaned.csv')
    co2_predictions = pd.read_csv('csv_files/predictions_scores/co2_predictions_with_clusters.csv')

    # Fix column names
    co2_data.columns = [col.strip() for col in co2_data.columns]
    co2_predictions.columns = [col.strip() for col in co2_predictions.columns]

    # Merge Country Names
    country_mapping = co2_data[['Country', 'Year', 'CO2_Emissions']]
    merged_predictions = co2_predictions.merge(country_mapping, on=['Country', 'Year'], how='left')

    # Model Selector
    model_selector = st.selectbox("Select Model for CO₂ Prediction", merged_predictions['Model'].unique())

    # Country Selector
    country_list = ['All'] + sorted(merged_predictions['Country'].unique())
    selected_country = st.selectbox("Select Country (optional)", country_list)

    # Filter only future years (2025-2030)
    future_predictions = merged_predictions[(merged_predictions['Year'] >= 2025) & (merged_predictions['Year'] <= 2030)]
    filtered_predictions = future_predictions[future_predictions['Model'] == model_selector]

    if selected_country != 'All':
        filtered_predictions = filtered_predictions[filtered_predictions['Country'] == selected_country]

    # Ensure positive size values
    filtered_predictions['Prediction_Positive'] = filtered_predictions['Prediction'].clip(lower=0)

    # Tabs: Map and Scores
    tab1, tab2 = st.tabs(["🌎 Map", "📈 Scores"])
```

```python
tab1, tab2 = st.tabs(["🌎 Map", "📈 Scores"])

with tab1:
    fig = px.scatter_geo(
        filtered_predictions,
        locations='Country',
        locationmode='country names',
        color='Prediction',
        hover_name='Country',
        size='Prediction_Positive',
        size_max=20,
        animation_frame='Year',
        projection='natural earth',
        color_continuous_scale=[[0, "green"], [0.5, "yellow"], [1, "red"]],
        range_color=[0, filtered_predictions['Prediction_Positive'].max()],
        title=f"CO₂ Emissions Prediction by Country ({model_selector}) (2025-2030)"
    )
    st.plotly_chart(fig, use_container_width=True)

with tab2:
    st.subheader("📈 Model R2 Scores by Country")
    r2_scores = merged_predictions[['Country', 'Model', 'R2_Score']].drop_duplicates()
    r2_scores = r2_scores[r2_scores['Model'] == model_selector]  # Filter based on selected model
    st.dataframe(r2_scores.reset_index(drop=True))

# Global CO2 Trend Line
st.subheader("📊 Global Average CO₂ Emissions Trend (Historical + Predicted)")

# Prepare global average
global_trend = merged_predictions.groupby('Year')['Prediction'].mean().reset_index()

fig_trend = px.line(
    global_trend,
    x='Year',
    y='Prediction',
    title="Global Average CO₂ Emissions Over Time",
    labels={'Prediction': 'Average CO₂ Emissions (tons per capita)', 'Year': 'Year'},
)
fig_trend.update_traces(mode="lines+markers")
fig_trend.update_layout(yaxis_title="Average CO₂ Emissions", xaxis_title="Year")
st.plotly_chart(fig_trend, use_container_width=True)
```

Conclusion



Summary

- This project explores the historical patterns and future projections of climate change with a particular focus on Canada, using machine learning models to predict temperature changes, sea ice loss, precipitation trends, and global CO_2 emissions. By analyzing data from 1968 to 2024 and forecasting trends for 2025–2030, I aim to provide a data-driven view that can support policymakers, researchers, and environmental organizations in making informed decisions about climate action. The project combines traditional machine learning models (Random Forest, Gradient Boosting) with time series forecasting (Prophet) and applies techniques like regularization, feature engineering, and hyperparameter tuning to improve model accuracy.
- Throughout the dashboard, users can interact with dynamic visualizations including animated maps, predictive trend lines, and global comparisons. Although the models achieved varying R^2 scores depending on the dataset (with better performance on CO_2 emissions and more challenges in precipitation forecasting), the overall predictions highlight significant climate risks. Particularly, Arctic sea ice loss and global temperature rises are critical areas of concern, while Canada's regional precipitation patterns show more complex, fluctuating trends. The project also demonstrates the value of combining multiple models and feature enhancements like GDP categories and region mappings for improving prediction performance.

Summary

- Thank you for reviewing this project. It was a rewarding experience to apply machine learning, data visualization, and climate science together in one platform.
- The whole Streamlit Dashboard and R2 score spreadsheets can be found in relative folders.

Thank You!

Student Name: Aysegul Dahi
Student Number: 300387536