

# BC Address Analyzer - Technical & User Manual

---

## Table of Contents

1. [Overview](#)
2. [System Requirements](#)
3. [Installation Guide](#)
4. [Test Data and Location](#)
5. [Canada Post AddressComplete API Configuration](#)
6. [Google Maps API Configuration](#)
7. [Application Features](#)
8. [Step-by-Step Usage Guide](#)
9. [Troubleshooting](#)
10. [Performance & Optimization](#)
11. [Data Privacy & Security](#)
12. [Maintenance & Updates](#)

## 1. Overview

The **BC Address Analyzer** is a comprehensive Streamlit web application designed for analyzing healthcare address data, specifically for precariously housed patients in British Columbia. The application provides advanced data processing, address elementization, descriptive analytics, visualization and modeling capabilities.

### Key Capabilities

- **Multi-table CSV data upload and management**
- **Address elementization using Canada Post AddressComplete API**
- **Healthcare data merging (discharge-to-visit records)**
- **Comprehensive descriptive analysis**
- **Advanced data visualization with 10 plot types**
- **Machine learning modeling frameworks**
- **Real-time data persistence and session management**

## 2. System Requirements

### Minimum Requirements

- **Operating System:** Windows 10+, macOS 10.14+, or Linux Ubuntu 18.04+
- **Python:** Version 3.9 or higher
- **RAM:** 4GB minimum, 8GB recommended
- **Storage:** 10GB free space
- **Internet:** Stable connection required for API calls

### 3. Installation Guide

#### Step 1: Environment Setup

```
# Create virtual environment (recommended)
python3 -m venv pehenv

# Activate environment
# On macOS/Linux:
source pehenv/bin/activate
# On Windows:
pehenv\Scripts\activate
```

#### Step 2: Install Dependencies

```
# Install required packages
pip install -r requirements.txt
```

#### Step 3: Launch Application

```
# Start the Streamlit application
streamlit run bc_addresses_analysis.py

# Optional: Use with tunnel for external access
streamlit run bc_addresses_analysis.py & npx localtunnel --port 8502
```

#### Required Python Packages

- **streamlit** - Web application framework
- **pandas** - Data manipulation and analysis
- **numpy** - Numerical computing
- **matplotlib** - Data visualization
- **seaborn** - Statistical data visualization
- **scikit-learn** - Machine learning library (includes Logistic Regression, RandomForest, preprocessing, metrics)
- **xgboost** - XGBoost gradient boosting framework
- **catboost** - CatBoost gradient boosting library
- **lightgbm** - LightGBM gradient boosting framework
- **geopy** - Geospatial calculations and distance computations
- **requests** - HTTP library for API calls
- **urllib3** - HTTP client for Python (used for Canada Post API)
- **xml** - XML parsing for API responses (built-in Python library)

## 4. Test Data and Location

### Sample Data Files

The application includes several test datasets located in the `/data` directory for testing and demonstration purposes:

#### Available Test Files

- **paris\_test\_dataset.csv** - Sample patient address data (Paris dataset format)
- **cerner1\_test\_dataset.csv** - Healthcare system data (Cerner dataset format #1)
- **cerner2\_test\_dataset.csv** - Healthcare system data (Cerner dataset format #2)
- **cerner3\_test\_dataset.csv** - Healthcare system data (Cerner dataset format #3)
- **non\_unique\_facilities.csv** - Non-unique / Long term care facilities with postal codes
- **facilities.csv** - ED facilities with postal code data
- **sample\_addresses.csv** - General address samples for testing
- **test\_records\_modeling.csv** - Machine learning modeling test dataset

#### Data File Descriptions

##### Healthcare Test Data

- **Cerner Files:** Contain simulated emergency patient discharge and visit records
  - **Columns:**
    - Patient IDs, admission/discharge dates
    - Address fields for elementization testing
    - Healthcare facility codes and locations

##### Facility Reference Data

- **facilities.csv:** ED facilities reference dataset
  - **Columns:** FacilityName, FacilityType, FacilityCategory, Address, PostalCode
- **non\_unique\_facilities.csv:** Non-unique / Long term care facilities reference dataset
  - **Columns:** FacilityName, FacilityType, FacilityCategory, Address, PostalCode


##### Address Data

- **sample\_addresses.csv:**
  - **Columns:** addressString
  - Various BC address formats
  - Mix of residential and commercial addresses
  - Useful for testing address elementization
- **paris\_test\_dataset.csv:** Patient address history dataset with comprehensive address tracking

- **Columns:** SourceAddressID, ClientID, IsAddressCurrent, IsHomeless, SourceCreateDate, SourceUpdateDate, StartDateID, EndDateID, AddressTypeID, HouseTypeID, City, SourceProvinceCode, SourcePostalCode, PostalCodeID, Country, AddressLine1, AddressLine2, AddressLine3
- **test\_records\_modeling.csv:** Machine learning modeling test dataset
  - **Purpose:** Designed specifically for testing ML modeling functionality
  - **Features:** Mixed data types (numerical and categorical) for comprehensive model testing
  - **Use Cases:** Classification and regression modeling examples, hyperparameter tuning validation, feature importance analysis

## Quick Start with Test Data

### Option 1: Using Individual/Bulk Files

1. Launch the BC Address Analyzer application
2. Go to  **Upload** tab
3. Select one or more test files from **/data** directory:
4. Review uploaded data in the preview section
5. Proceed with processing and analysis

### Option 3: End-to-End Workflow Test

```
# Complete test sequence:  
1. Upload: cerner1_test_dataset.csv  
2. Preprocess: Run address elementization  
3. Upload: cerner2_test_dataset.csv and cerner3_test_dataset.csv (for  
Merge Discharge to emergency visit records)  
4. Select EDVisitDate column for Table3 and DischargeDate column from  
Table3  
5. Adjust the Date matching tolerance (days) and choose columns to include  
in final merged table.  
6. Click: Merge Tables  
7. Analyze: Run all descriptive analysis sections  
8. Visualize: Create plots with processed data  
9. Model: Test machine learning features
```

## Test Data Specifications

### Expected Data Format

All CSV test files follow this general structure:

- **Patient ID columns:** ClientID, VisitID, SourceAddressID or similar
- **Date columns:** SourceCreateDate, SourceUpdateDate, StartDateID, EndDateID, DischargeDate, EDVisitDate

- **Address columns:** AddressLine1, AddressLine2, AddressLine3, StreetAddress1, StreetAddress2, City, PostalCode, Country

### Data Quality Notes

- Test data includes intentional missing values for testing
- Some addresses may be incomplete to test postal codes retrieving
- Facility datasets includes both ED facility and non-unique / long term care postal codes
- Date ranges span multiple years for temporal analysis testing

### Testing Checklist

- ☐ Data uploads successfully without errors
- ☐ Address elementization processes correctly
- ☐ Date parsing works for temporal analysis
- ☐ Facility matching identifies correct associations
- ☐ Visualizations generate without issues
- ☐ All analysis sections complete successfully

### File Locations

### Project Structure

```
PEH-cohort/
├── bc_addresses_analysis.py          # Main application
├── requirements.txt                  # Dependencies
├── setup.ipynb                      # Setup notebook
├── BC_Address_Analyzer_Instructions.md # This documentation
├── BC_Address_Analyzer_Instructions.pdf # This documentation (PDF
Version)
└── data/                            # Test data directory
    ├── paris_test_dataset.csv
    ├── cerner1_test_dataset.csv
    ├── cerner2_test_dataset.csv
    ├── cerner3_test_dataset.csv
    ├── facilities.csv
    ├── non_unique_facilities.csv
    └── sample_addresses.csv
```

### Access Permissions

- Ensure read permissions for all data files
- Verify file paths are accessible from application directory
- Check file encoding matches system requirements (UTF-8 recommended)

## 5. Canada Post AddressComplete API Configuration

### API Key Management

The application uses Canada Post's AddressComplete API for address elementization. **This is the most critical configuration step.**

### Current API Key Location

- **File:** `bc_addresses_analysis.py`
- **Line:** ~453
- **Current Key:** `HW69-DD53-MU77-UY93`

```
# IMPORTANT: Update this API key when needed
api_key = "HW69-DD53-MU77-UY93"
```

### How to Change API Key

#### 1. Obtain New API Key:

- Visit [Canada Post Developer Portal](#)
- Register for AddressComplete API access
- Obtain your unique API key

#### 2. Update Application Code:

```
# Find line ~453 in bc_addresses_analysis.py
# Replace the existing key with your new key:
api_key = "YOUR-NEW-API-KEY-HERE"
```

#### 3. Alternative: Environment Variable (Recommended for Production):

```
# Modify the code to use environment variables:
import os
api_key = os.getenv("CANADA_POST_API_KEY", "ZX12-HA39-BE19-ZZ84")
```

### API Usage Limits

- **Rate Limit:** Avoid making more than 10 requests per second
- **Daily Limits:** Check your subscription limits (Free Trial : Maximun 100 transactions per day)
- **Fallback:** Application includes manual address parsing when API fails

### API Endpoints Used

1. **Find Address:** `http://ws1.postescanada-canadapost.ca/AddressComplete/Interactive/Find/v2.10/xmla.ws`
2. **Retrieve Address:** `http://ws1.postescanada-canadapost.ca/AddressComplete/Interactive/Retrieve/v2.11/xmla.ws`

## 6. Google Maps API Configuration

The application can optionally use Google Maps Geocoding API for coordinate generation from postal codes.

### Google Maps API Key Setup

- **File:** `bc_addresses_analysis.py`
- **Line:** ~192
- **Current Setting:** `API_KEY = ''`

### How to Configure Google Maps API

#### 1. Obtain Google Maps API Key:

- Visit [Google Cloud Console](#)
- Create a new project or select existing project
- Enable the "Geocoding API" service
- Generate an API key

#### 2. Update Application Code:

```
# Find line ~192 in bc_addresses_analysis.py
# Replace the empty string with your API key:
API_KEY = 'YOUR_GOOGLE_MAPS_API_KEY_HERE'
```

#### 3. API Security (Recommended):

- Restrict API key to Geocoding API only
- Set up billing limits to control costs
- Monitor usage in Google Cloud Console

### API Usage Limits & Costs

- **Free Tier:** 10,000 free geocoding requests/month (Essentials SKU)
- **Rate Limiting:** Application includes 0.2 second delays between requests
- **Cost Management:** Set billing alerts and daily quotas
- **Caching:** Coordinates are cached to avoid duplicate API calls

### API Endpoint Used

- **Geocoding API:** <https://maps.googleapis.com/maps/api/geocode/json>

## 7. Application Features

### Multi-Tab Interface

#### 1. Upload Tab ()

- **CSV File Upload:** Multiple file support with drag-and-drop
- **Data Preview:** Interactive table display with sorting/filtering
- **Table Management:** Rename, delete, and organize uploaded datasets
- **Format Validation:** Automatic CSV structure verification

## 2. Data Preprocessing Tab ()

- **Address Elementization:** Convert full addresses into structured components
- **Column Selection:** Choose address fields for processing
- **API Integration:** Real-time Canada Post API calls
- **Fallback Parsing:** Manual address parsing when API unavailable
- **Google Maps Coordinate Generation:** Convert postal codes to latitude/longitude coordinates using Google Maps Geocoding API
- **Discharge-to-Visit Record Merging:** Combine healthcare records with configurable date tolerance and column selection

## 3. Descriptive Analysis Tab ()

- **Move Frequency Relative to Patient Analysis:** Track patient address changes over time
- **Frequency of Missing or Incomplete Address Fields Analysis:** Identify data quality issues
- **Residential vs Commercial Classification:** Categorize address types
- **ED Facility Matching:** Match patient addresses with ED Facilities and Non-unique / Long term care Facilities using Postal codes
- **Distance Moved Between Address Changes:** Calculate geographic distances between consecutive addresses using coordinate data
- **Number of Patients Sharing Same Address Analysis:** Identify multiple patients at same addresses

## 4. Visualization Tab ()

- **Table Selection:** Choose datasets for visualization
- **Column Selection:** Multi-select visualization variables
- **10 Plot Types Available:**
  - Bar Charts (count and grouped)
  - Line Charts (time series and trends)
  - Scatter Plots (correlation analysis)
  - Box Plots (distribution analysis)
  - Histograms (frequency distributions)
  - Pie Charts (categorical proportions)
  - Heatmaps (correlation matrices)
  - Count Plots (categorical frequencies)
  - Distribution Plots (statistical distributions)
  - Pair Plots (multi-variable relationships)
- **Interactive Controls:** Dynamic axis selection and filtering
- **Export Options:** Save plots as PNG/PDF

## 5. Modeling Tab ()

- **Dataset Selection:** Choose from uploaded datasets for modeling



- **Target Variable Selection:** Select dependent variable (automatic problem type detection)
- **Feature Engineering:** Multi-select independent variables with preprocessing
- **Model Types:** Logistic Regression, RandomForest, XGBoost, CatBoost, LightGBM
- **Hyperparameter Tuning:** Manual parameter setting or automated Grid Search CV
- **Train-Test Split:** Configurable test size and random state
- **Model Evaluation:** Comprehensive performance metrics including:
  - Classification: Accuracy, Precision, Recall, F1-Score, Confusion Matrix, Classification Report
  - Regression:  $R^2$  Score, MSE, MAE, RMSE, Residual Plots, Actual vs Predicted plots
- **Feature Importance:** Visual and tabular feature importance analysis (when available)
- **Results Export:** Save predictions, model info, and download as CSV files
- **Session Persistence:** All model results saved to session state for further analysis


## 8. Step-by-Step Usage Guide

### Getting Started


#### Step 1: Launch Application

1. Open terminal/command prompt
2. Navigate to project directory
3. Activate virtual environment
4. Run: `streamlit run bc_addresses_analysis.py`
5. Open browser to `http://localhost:8501`


#### Step 2: Upload Your Data

1. Go to  **Upload** tab
2. Click "Choose files" or drag CSV files
3. Review uploaded tables in the preview
4. Rename tables if needed using the rename functionality



#### Step 3: Process Addresses (Optional but Recommended)

1. Go to  **Data Preprocessing** tab
2. Select table containing address data
3. Choose address columns (street, city, postal code, etc.)
4. Check "Elementize the Addresses?"
5. Wait for processing (uses Canada Post API)
6. Review elementized results with E\_ prefixed columns


#### Step 3.5: Generate Coordinates (Optional - Enables Distance Analysis)

1. **Prerequisites:** Ensure Google Maps API key is configured (line ~192)
2. Stay in  **Data Preprocessing** tab
3. Select table with postal code data (elementized table recommended)
4. Check "Get coordinate from an address postcode? (via Google API)"
5. System processes unique postal codes and adds Latitude/Longitude columns
6. Review coordinate data in updated table


#### Step 3.7: Calculate Distance Movements (Optional - Requires Coordinates)

1. Go to  **Descriptive** analysis tab
2. Select table with coordinate data
3. Check "  Distance Moved Between Address Changes"
4. Choose appropriate date column for chronological ordering
5. Review distance calculations and summary statistics
6. Results include MoveDistance(KM) column with distances between consecutive addresses

#### Step 4: Perform Analysis

1. Go to  **Descriptive** tab
2. Select desired analysis type:
  - **Move Frequency:** Track patient mobility
  - **Missing Fields:** Data quality assessment
  - **Residential Classification:** Address categorization
  - **ED Facility Matching:** Healthcare facility analysis
  - **Address Sharing:** Multi-patient address identification
3. Results automatically save back to original tables

## Step 5: Create Visualizations


1. Go to  **Visualization** tab
2. Select table for visualization
3. Choose columns to include
4. Select plot type from 10 available options
5. Configure plot parameters (axes, grouping, etc.)
6. Generate and export visualizations

## Advanced Features

### Discharge-to-EDVisit Record Merging

1. Upload both emergency visit record (cerner2) and discharge record (cerner3) CSV files
2. In Data Preprocessing tab, scroll to "Merge Discharge to ED Visit Records"
3. Select discharge table and visit table
4. Configure date tolerance
5. Confirm the new merged table name
6. Select columns to include in final merged table
7. Merge tables
8. Review merged results

### Google Maps Coordinate Generation

1. **Prerequisites:**
  - Add your Google Maps API key to line ~192 in `bc_addresses_analysis.py`
  - Ensure your dataset contains a `PostalCode` column
  - Have elementized addresses (recommended) or postal code data
2. **Process:**
  - Go to  **Data Preprocessing** tab
  - Select table with postal code data
  - Check "Get coordinate from an address postalcode? (via Google API)"
  - System will automatically detect postal codes and generate coordinates
  - New `Latitude` and `Longitude` columns will be added to your dataset
3. **Features:**
  - Rate limiting (0.2 second delays) to respect Google API limits
  - Error handling for invalid postal codes
  - Coordinate caching to avoid duplicate API calls

#### 4. Google Maps API Setup:


- Visit [Google Cloud Console](#)
- Enable the Geocoding API
- Create an API key and update `API_KEY = 'YOUR_KEY_HERE'` in the code
- Set usage limits to manage costs

### Distance Moved Between Address Changes

#### 1. Prerequisites:

- Coordinate generation must be completed first (requires Latitude/Longitude columns)
- Dataset must contain date columns for chronological ordering
- Patient data with multiple address records per ClientID

#### 2. Process:

- Go to  **Descriptive** analysis tab
- Select table with coordinate data
- Check "📍 Distance Moved Between Address Changes"
- Select appropriate date column for chronological sorting of addresses
- System calculates distances between consecutive addresses for each patient

#### 3. Analysis Features:

- Uses `geopy.distance.geodesic` for accurate geographic distance calculations
- Adds `MoveDistance(KM)` column with distances in kilometers

#### 4. Use Cases:

- Analyze patient mobility patterns
- Identify patients with frequent long-distance moves
- Study geographic stability of precariously housed populations
- Calculate total distance traveled across all address changes

### ED Facility Matching

1. Prepare ED facility CSV files (primary and secondary)
2. Upload patient data with postal codes
3. In Descriptive tab, check "Check ED Visit Centers"
4. Upload facility files when prompted
5. Review matching results showing facility associations

### Independent Table Analysis

- Each analysis section allows independent table selection
- Results persist back to original selected tables
- No cross-contamination between analyses
- Session state maintains all changes

## 9. Troubleshooting

### Common Issues

#### "API Key Invalid" Error

**Problem:** Canada Post API returns authentication error **Solution:**

1. Verify API key is correct in line ~453
2. Check API key hasn't expired
3. Ensure internet connection is stable
4. Contact Canada Post if key issues persist

#### "Google Maps API Error"

**Problem:** Coordinate generation fails or returns errors **Solution:**

1. Verify Google Maps API key is correctly set in line ~192
2. Ensure Geocoding API is enabled in Google Cloud Console
3. Check API key hasn't exceeded daily/monthly limits
4. Verify billing is set up (required for Google Maps API)
5. Check postal code format (Canadian format: A1A 1A1)
6. Monitor Google Cloud Console for error details

#### "PostalCode Column Not Found"

**Problem:** Coordinate generation requires PostalCode column **Solution:**

1. Ensure dataset contains a column named 'PostalCode'
2. Run address elementization first to generate postal codes
3. Check column naming - must be exact match 'PostalCode'
4. Verify data contains valid Canadian postal codes

#### "Latitude/Longitude Columns Required"

**Problem:** Distance calculation missing coordinate data **Solution:**

1. Run coordinate generation first using Google Maps API
2. Ensure 'Latitude' and 'Longitude' columns exist in dataset
3. Check that coordinate generation completed successfully
4. Verify coordinate data is not all null/empty

#### "Missing Columns" Error

**Problem:** Analysis requires specific column names **Solution:**

1. Check required columns for each analysis type
2. Rename columns to match expected format
3. Ensure data preprocessing completed successfully

4. Review column mapping requirements

### "Memory Error" During Processing

**Problem:** Large datasets cause memory issues **Solution:**

1. Process smaller data chunks
2. Increase system RAM
3. Close unnecessary applications
4. Use data filtering to reduce dataset size

### "API Rate Limit Exceeded"

**Problem:** Too many API calls in short time **Solution:**

1. Reduce processing frequency
2. Implement delays between API calls
3. Process data in smaller batches
4. Consider upgrading API subscription

### Visualization Not Displaying

**Problem:** Plots fail to render **Solution:**

1. Check selected columns have appropriate data types
2. Verify no empty datasets
3. Ensure matplotlib backend compatibility
4. Refresh browser and retry

### Technical Support

#### Log Files

- Streamlit logs: Check terminal output
- Application errors: Review browser console
- API responses: Monitor network tab in browser tools

#### Data Validation

- CSV format: Ensure proper encoding (UTF-8)
- Column names: No special characters or spaces
- Date formats: Use YYYY-MM-DD or MM/DD/YYYY consistently

## 10. Performance & Optimization

### Best Practices

#### Data Management

- **File Size:** Keep CSV files under 100MB for optimal performance
- **Column Count:** Limit to essential columns for analysis
- **Data Types:** Ensure proper data type assignment
- **Missing Values:** Handle nulls before processing

#### API Usage

- **Batch Processing:** Process addresses in batches of 50-100
- **Caching:** Results are automatically cached in session state
- **Fallback:** Manual parsing reduces API dependency
- **Monitoring:** Track API usage to avoid limits

#### Memory Management

- **Session State:** Clear unnecessary variables periodically
- **Large Datasets:** Use data sampling for initial exploration
- **Visualization:** Limit plot data points for responsiveness

#### Session Management

- Automatic data persistence across tabs
- Table state maintained throughout session
- Results automatically saved to original tables

## 11. Data Privacy & Security

### Security Considerations

#### Data Handling

- **Local Processing:** All data processed locally on your machine
- **No Data Storage:** Application doesn't store data permanently
- **Session Isolation:** Each browser session is independent
- **API Security:** Uses HTTPS for Canada Post API calls

#### Privacy Compliance

- **Local Deployment:** Run on secure networks for sensitive data
- **Data Anonymization:** Consider removing patient identifiers when possible
- **Access Control:** Implement user authentication for production use

## Best Practices

- Use secure networks for data processing
- Regularly update dependencies for security patches
- Monitor API usage logs
- Implement data backup procedures

## 12. Maintenance & Updates

### Regular Maintenance

#### Weekly Tasks

- ☐ Check API key validity and usage limits
- ☐ Review application logs for errors
- ☐ Update dependencies if security patches available
- ☐ Backup important analysis results

#### Monthly Tasks

- ☐ Update Python packages: `pip install -r requirements.txt --upgrade`
- ☐ Review Canada Post API documentation for changes
- ☐ Check Streamlit version compatibility
- ☐ Performance monitoring and optimization

#### Quarterly Tasks

- ☐ Full dependency audit and updates
- ☐ API key renewal if needed
- ☐ User feedback review and feature planning
- ☐ Security assessment and updates

### Update Procedures

#### Application Updates

1. Backup current working version
2. Test updates in development environment
3. Update requirements.txt if needed
4. Deploy to production after testing

#### API Updates

1. Monitor Canada Post API changelog
2. Test API changes in development
3. Update API endpoint URLs if needed
4. Update error handling for new API responses



## Support Resources

### Documentation

- **Streamlit Docs:** [docs.streamlit.io](https://docs.streamlit.io)
- **Pandas Docs:** [pandas.pydata.org](https://pandas.pydata.org)
- **Canada Post API:** [developer.canadapost.ca](https://developer.canadapost.ca)

### Community Support

- **Streamlit Forum:** [discuss.streamlit.io](https://discuss.streamlit.io)
- **Stack Overflow:** Tag questions with `streamlit`, `pandas`
- **GitHub Issues:** Report bugs in project repository

---

## Quick Reference

### Essential File Locations

- **Main Application:** `bc_addresses_analysis.py`
- **Dependencies:** `requirements.txt`
- **Setup Script:** `setup.ipynb`
- **Documentation:** `BC_Address_Analyzer_Instructions.md`

### Key Configuration Points

- **Google API Key:** Line ~192 in `bc_addresses_analysis.py`
- **Canada Post API Key:** Line ~453 in `bc_addresses_analysis.py`
- **Port:** Default 8501 (configurable)
- **Memory Limits:** Adjust based on system capacity
- **Theme:** Toggle in application sidebar

---

*Last Updated: 27 August 2025*

*Version: 1.0*

*Compatibility: Python 3.9+, Streamlit 1.25+*