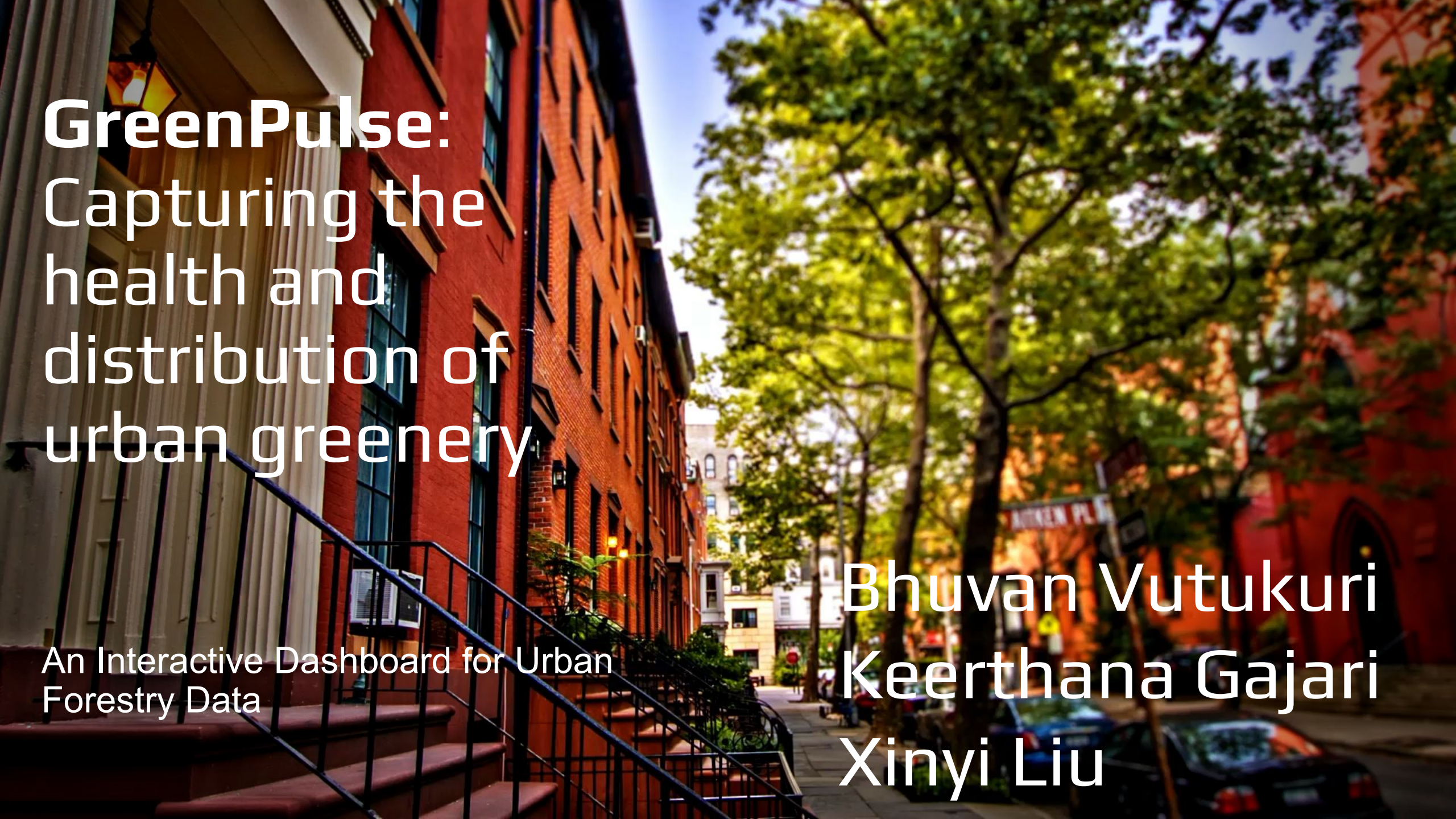


GreenPulse: Capturing the health and distribution of urban greenery

An Interactive Dashboard for Urban Forestry Data

Bhuvan Vutukuri
Keerthana Gajari
Xinyi Liu

Introduction

- Urban trees play a critical role in enhancing ecological and aesthetic value.
- Effective monitoring of tree health and distribution is essential for sustainable urban planning.
- This project provides an interactive platform to explore urban forestry data.

The Problem

- Urban forests are under pressure due to limited resources for monitoring and management.
- Key challenges include:
  - Assessing tree health across neighborhoods.
  - Understanding the spatial distribution of trees.
  - Informing policymakers with actionable insights.

Objectives

1. Explore tree health, size, and spatial distribution.
2. Provide insights into tree conditions across different neighborhoods.
3. Develop interactive visualizations for data exploration and analysis.

Data Overview

- **Source**: NYC OpenData.
- **Key Attributes**:
  - tree_dbh, stump_diam: Tree diameter and Stump diameter.
  - status, health: Tree status(alive or dead) and health(good, fair or poor).
  - spc_common: Species common name of the tree.
  - borough: Boroughs in which tree is located.
  - postcode and zip_city: Zip code and city in which tree is located.
  - latitude and longitude: Geographic coordinates of the tree.

Note:

We extracted only 5% of the entire dataset to address computational limitations resulting in 34k rows.

Methodology

- **Data Preparation**:
  - Cleaned and processed raw tree census data.
  - Handled missing values and outliers.
- **Visualization Development**:
  - Built tabular and interactive map visualizations.
- **Data Analysis**:
  - Aggregated data to explore trends across health, diameter, and boroughs.

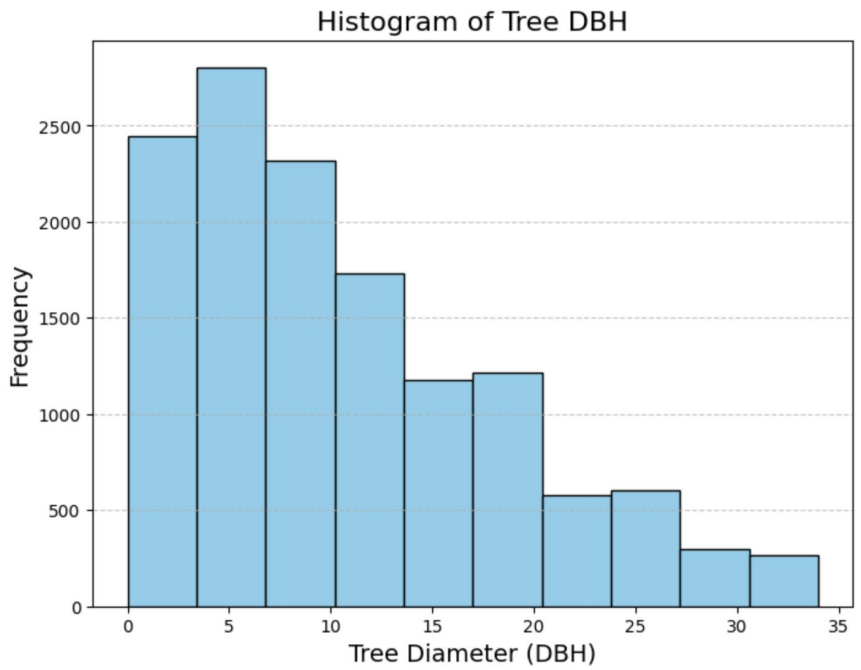
Data Cleaning

```
[4] print(df.shape)
    print(df.columns)

    (34189, 45)
    Index(['tree_id', 'block_id', 'created_at', 'tree_dbh', 'stump_diam',
           'curb_loc', 'status', 'health', 'spc_latin', 'spc_common', 'steward',
           'guards', 'sidewalk', 'user_type', 'problems', 'root_stone',
           'root_grate', 'root_other', 'trunk_wire', 'trnk_light', 'trnk_other',
           'brch_light', 'brch_shoe', 'brch_other', 'address', 'postcode',
           'zip_city', 'community board', 'borocode', 'borough', 'cncldist',
           'st_assem', 'st_senate', 'nta', 'nta_name', 'boro_ct', 'state',
           'latitude', 'longitude', 'x_sp', 'y_sp', 'council district',
           'census tract', 'bin', 'bbl'],
          dtype='object')
```

```
Missing values after cleaning:
tree_id             0
block_id            0
created_at          0
tree_dbh            0
stump_diam          0
curb_loc            0
status              0
health              0
spc_common          0
root_stone          0
root_grate          0
root_other          0
trunk_wire          0
trnk_light          0
trnk_other          0
brch_light          0
brch_shoe           0
brch_other          0
address             0
postcode            0
zip_city            0
community board     0
borough             0
cncldist            0
st_assem            0
st_senate           0
nta                 0
nta_name            0
latitude            0
longitude           0
```

```python
# 1. Histogram of tree_dbh
plt.figure(figsize=(8, 6))
plt.hist(df['tree_dbh'], bins=10, color='skyblue', edgecolor='black')
plt.title('Histogram of Tree DBH', fontsize=16)
plt.xlabel('Tree Diameter (DBH)', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Histogram of Tree Diameter

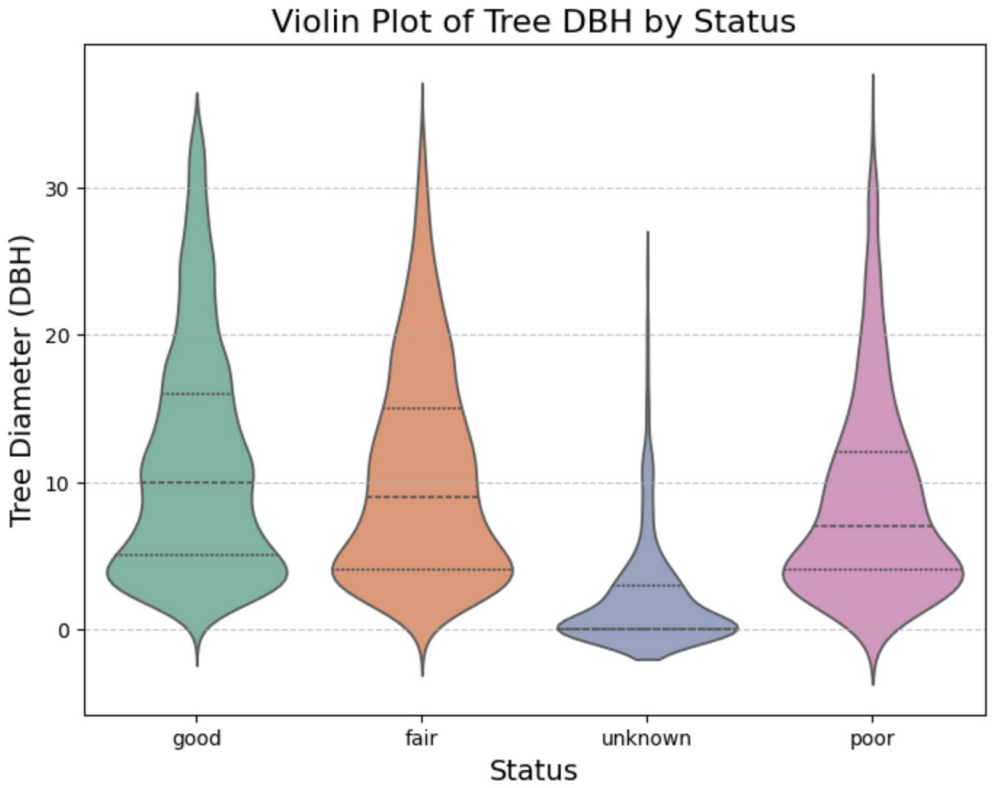
- **Histogram Overview**
- Histogram visualizes the distribution of tree diameters (tree_dbh) across the dataset.
- Groups diameter values into intervals (bins) and shows how frequently trees fall into each range.
- **Key Observations**
- Most trees have diameters within the range of 5 to 10.
- Larger tree diameters are less common, indicating fewer mature trees in the dataset.

```python
#2. Violin plot of the distribution of tree_dbh based on status
plt.figure(figsize=(8, 6))
sns.violinplot(x='health', y='tree_dbh', data=df, palette='Set2', inner='quartile')
plt.title('Violin Plot of Tree DBH by Status', fontsize=16)
plt.xlabel('Status', fontsize=14)
plt.ylabel('Tree Diameter (DBH)', fontsize=14)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
<ipython-input-8-81fb4e6f88bb>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.violinplot(x='health', y='tree_dbh', data=df, palette='Set2', inner='quartile')
```



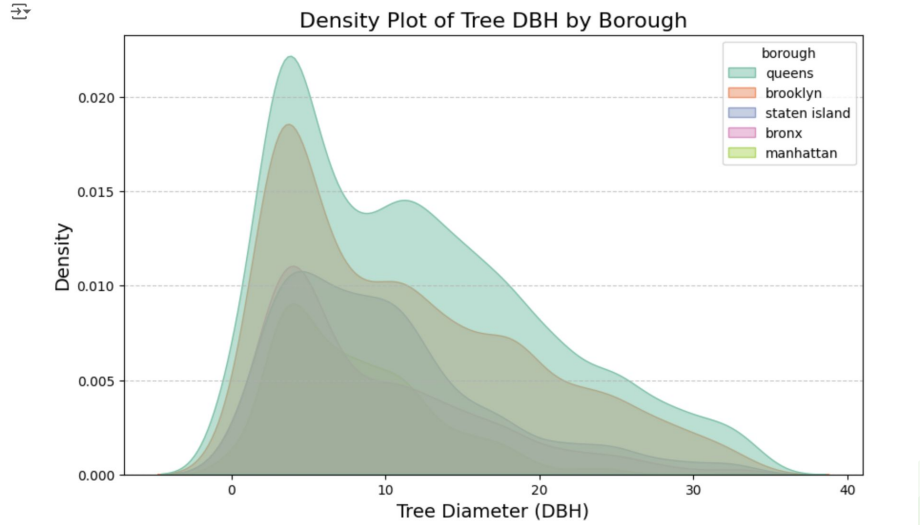
Violin Plot of Tree Diameter by Health Status

Violin plot is used to visualize the distribution of tree diameters (tree_dbh) for each health status category (Good, Fair, Poor, unknown).

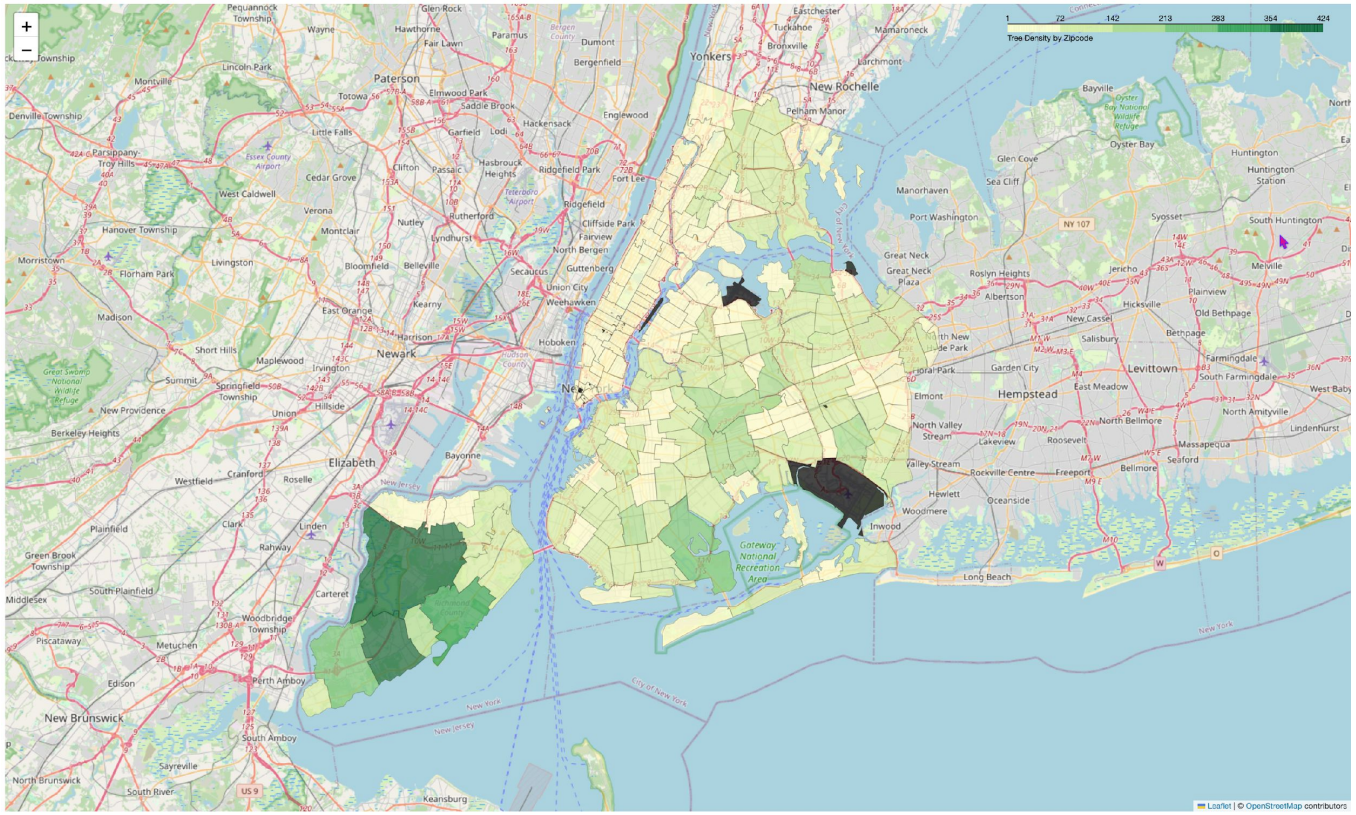
Density Plot of Tree Diameter by borough

- **Density Plot Overview**
- This density plot compares the distribution of tree diameters (tree_dbh) across different boroughs.
- Each borough is represented by a separate curve, showing the probability density of tree sizes.
- **Key Observations**
- **Borough Comparisons**:
    1. Queens has the highest density for smaller trees, suggesting a younger tree population.
    2. Staten island shows a wider spread, indicating a mix of young and mature trees.

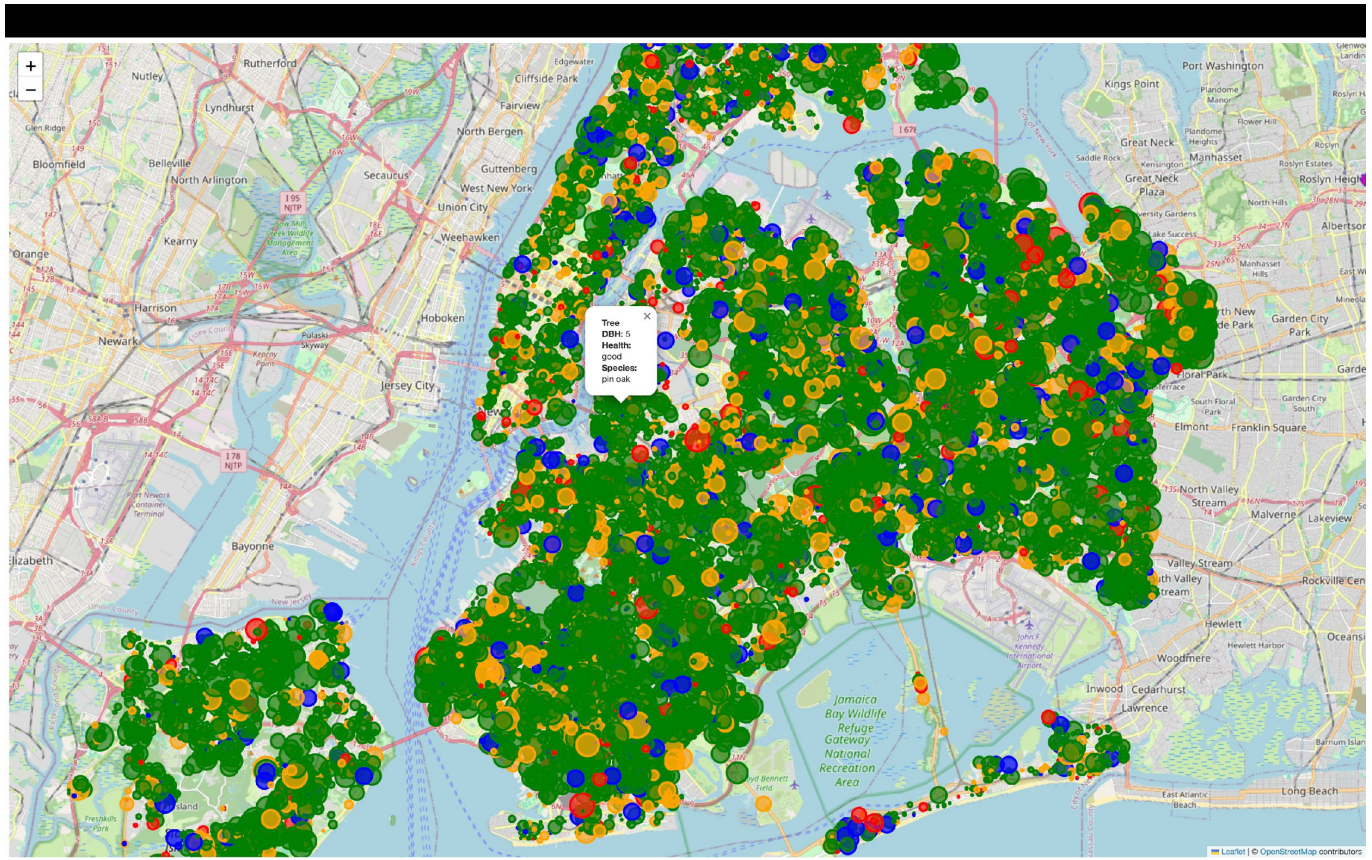
```
[ ] # 3. Grouped Density Plot: Tree Diameter by Borough
    plt.figure(figsize=(10, 6))
    sns.kdeplot(data=df, x='tree_dbh', hue='borough', fill=True, alpha=0.5, palette='Set2')
    plt.title('Density Plot of Tree DBH by Borough', fontsize=16)
    plt.xlabel('Tree Diameter (DBH)', fontsize=14)
    plt.ylabel('Density', fontsize=14)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
```



Choropleth Map of Tree Density by Zip Code




```
#code of choropleth map of tree density by zipcode using folium
df['postcode'] = pd.to_numeric(df['postcode'], errors='coerce')
tree_density = df.groupby('postcode').size().reset_index(name='tree_count')
latitude = df['latitude'].mean()
longitude = df['longitude'].mean()
m = folium.Map(location=[latitude, longitude], zoom_start=11)
Choropleth(
    geo_data=geojson_path,
    name='choropleth',
    data=tree_density,
    columns=['postcode', 'tree_count'],
    key_on='feature.properties.postalCode',
    fill_color='YlGn',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='Tree Density by Zipcode'
).add_to(m)
m
```




```python
#bubble map having bubble size as tree dbh and color for health
df['health'].unique()
def health_to_color(health):
    color_map = {
        'good': 'green',
        'fair': 'orange',
        'poor': 'red'
    }
    return color_map.get(health, 'blue')
m = folium.Map(location=[df['latitude'].mean(), df['longitude'].mean()], zoom_start=12)
for _, row in df.iterrows():
    folium.CircleMarker(
        location=(row['latitude'], row['longitude']),
        radius=row['tree_dbh'] / 2,
        color=health_to_color(row['health']),
        fill=True,
        fill_color=health_to_color(row['health']),
        fill_opacity=0.7,
        popup=(
            f"<b>Tree DBH:</b> {row['tree_dbh']}<br>"
            f"<b>Health:</b> {row['health']}<br>"
            f"<b>Species:</b> {row['spc_common']}"
        )
    ).add_to(m)
m
```

Bubble Map of Tree Diameter, Health Status and Species name

Tabular Data Visualization

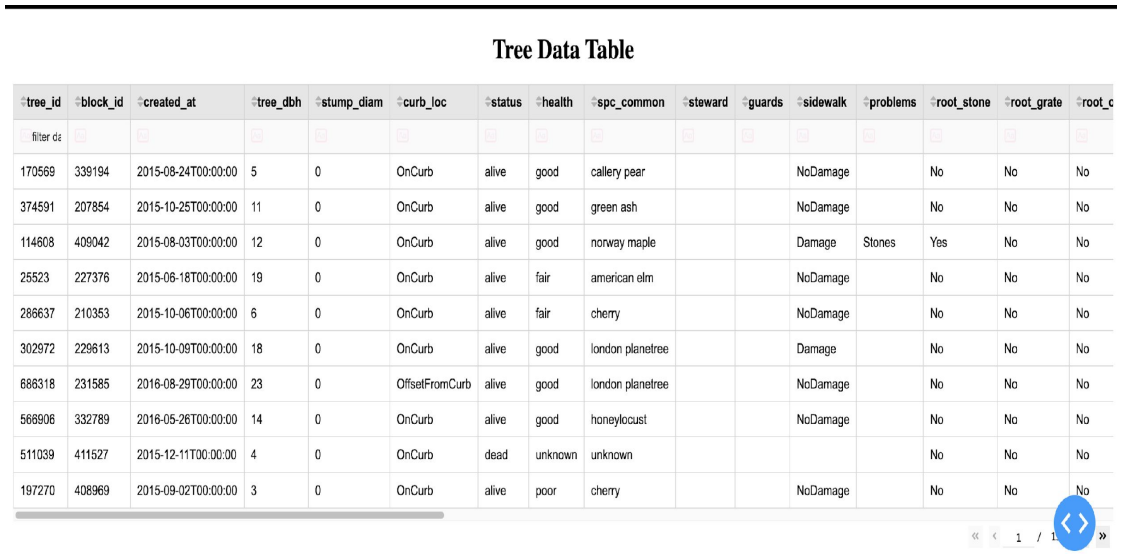
- **Tabular Data Overview**
- A tabular data visualization provides a detailed, row-by-row view of the dataset.
- It allows users to inspect and interact with individual tree records, including:
    - **Tree ID**: Unique identifier for each tree.
    - **Borough**: Location of the tree.
    - **Health**: Status of tree health (Good, Fair, Poor).
    - **Tree Diameter** (tree_dbh): Size of the tree.
    - **Problems**: Observed issues, such as root or branch damage.
- **Key Features**

1. **Interactive Filters**:
    1. Filter data by borough, health, or other attributes for focused analysis.
2. **Sorting**:
    1. Sort rows by tree diameter or health to identify trends easily.
3. **Download Capability**:
    1. Export filtered data for further offline analysis.

```python
# Initialize Dash app
app = Dash(__name__)

# Layout with Dash DataTable
app.layout = html.Div([
    html.H1("Tree Data Table", style={"textAlign": "center"}),
    dash_table.DataTable(
        id='tree-table',
        columns=[{"name": i, "id": i} for i in df.columns],
        data=df.to_dict('records'),
        style_table={'overflowX': 'auto', 'margin': 'auto'},
        style_cell={
            'textAlign': 'left',
            'padding': '10px',
            'fontFamily': 'Arial, sans-serif',
        },
        style_header={
            'backgroundColor': 'rgb(230, 230, 230)',
            'fontWeight': 'bold'
        },
        page_size=10,
        sort_action="native",
        filter_action="native",
    )
])

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```



Tree Data Table

tree_id	block_id	created_at	tree_dbh	stump_diam	curb_loc	status	health	spc_common	steward	guards	sidewalk	problems	root_stone	root_grate	root_c
filter da															
170569	339194	2015-08-24T00:00:00	5	0	OnCurb	alive	good	callery pear			NoDamage		No	No	No
374591	207854	2015-10-25T00:00:00	11	0	OnCurb	alive	good	green ash			NoDamage		No	No	No
114608	409042	2015-08-03T00:00:00	12	0	OnCurb	alive	good	norway maple			Damage	Stones	Yes	No	No
25523	227376	2015-06-18T00:00:00	19	0	OnCurb	alive	fair	american elm			NoDamage		No	No	No
286637	210353	2015-10-06T00:00:00	6	0	OnCurb	alive	fair	cherry			NoDamage		No	No	No
302972	229613	2015-10-09T00:00:00	18	0	OnCurb	alive	good	london planetree			Damage		No	No	No
686318	231585	2016-08-29T00:00:00	23	0	OffsetFromCurb	alive	good	london planetree			NoDamage		No	No	No
568906	332789	2016-05-26T00:00:00	14	0	OnCurb	alive	good	honeylocust			NoDamage		No	No	No
511039	411527	2015-12-11T00:00:00	4	0	OnCurb	dead	unknown	unknown					No	No	No
197270	408969	2015-09-02T00:00:00	3	0	OnCurb	alive	poor	cherry			NoDamage		No	No	No

« ‹ 1 / 1 › »

Aggregation Visualization of Tree Data

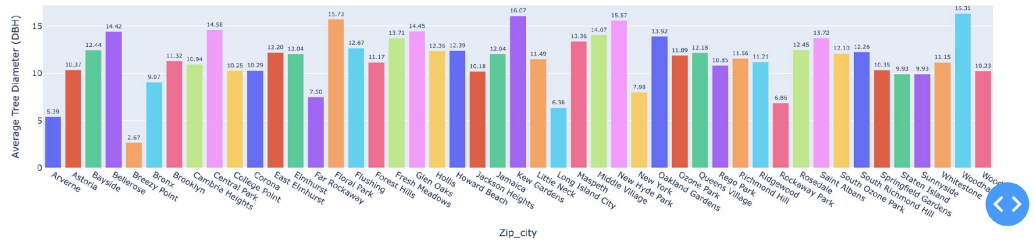
- **Aggregation Visualization Overview**
  - Aggregation visualizations summarize tree data by grouping it into meaningful categories (e.g., by borough or city).
  - These visualizations provide insights into trends and patterns that may not be apparent in individual records.
- **Key Features**
  - **Grouped Analysis**: Average tree diameter (tree_dbh) by borough highlights variations in tree size across regions.
  - **Bar Chart Representation**: Bar heights indicate average values, making comparisons intuitive and easy to interpret.
  - **Interactive Options**: Ability to select categories for customized aggregation.

Tree Data Visualization

Group By: City (Zip City)

Select Plot Type: Bar Plot

Average Tree Diameter by Zip_city

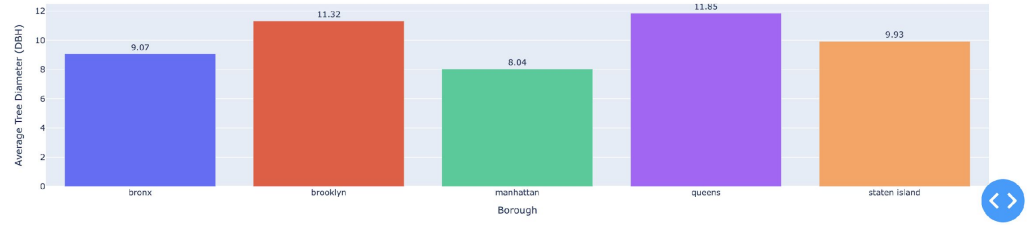


Tree Data Visualization

Group By: Borough

Select Plot Type: Bar Plot

Average Tree Diameter by Borough

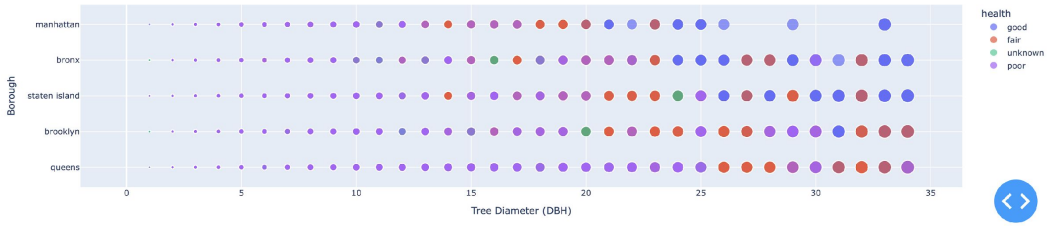


Tree Data Visualization

Group By: Borough

Select Plot Type: Scatter Plot

Scatter Plot of Tree DBH by Borough

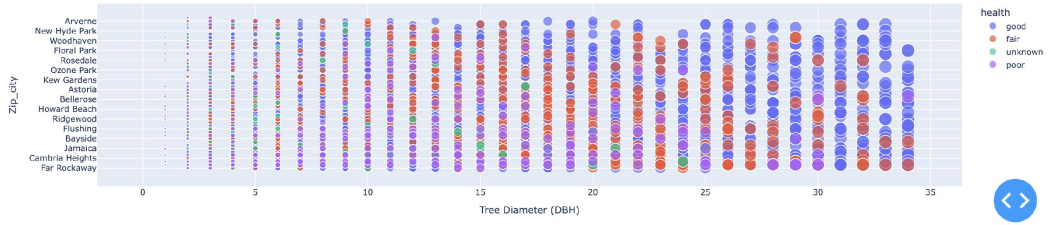


Tree Data Visualization

Group By: City (Zip City)

Select Plot Type: Borough / City (Zip City)

Scatter Plot of Tree DBH by Zip_city



Conclusion

- **Recap:**
  - The importance of urban forestry analysis.
  - Spatial distribution, health analysis.
- **Data Sources**
  - NYC OpenData: https://data.cityofnewyork.us/Environment/2015-Street-Tree-Census-Tree-Data/uvpi-gqnh/about_data
  - GeoJson data: https://github.com/fedhere/PUI2015_EC/blob/master/mam1612_EC/nyc-zip-code-tabulation-areas-polygons.geojson
- **Tools and Libraries:**
  - Python libraries: Pandas, Plotly, Dash.
  - mapping platforms: Folium.

Thank you