Project 4: K-Means, DBSCAN, and Mean Shift Clustering

Esther Bilenkin, Rushabh Parikh, and Stephanie Bravo-Heras

New York Institute of Technology

Table of Contents

[Introduction and Objectives 4](#_Toc185172187)

[Description of Data Sets 4](#_Toc185172188)

[Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data 4](#_Toc185172189)

[Data Set 2: Bicycle Parking 5](#_Toc185172190)

[Design and Methods 6](#_Toc185172191)

[Approach 6](#_Toc185172192)

[Data Cleaning 7](#_Toc185172193)

[Attribute Selection 9](#_Toc185172194)

[Data Training 13](#_Toc185172195)

[Results 16](#_Toc185172196)

[Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data 16](#_Toc185172197)

[Data Set 2: Bicycle Parking 18](#_Toc185172198)

[Conclusions and Future Works 21](#_Toc185172199)

[Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data 21](#_Toc185172200)

[Data Set 2: Bicycle Parking 21](#_Toc185172201)

[Team Member’s Contributions 22](#_Toc185172202)

[Group Activities: 22](#_Toc185172203)

[Rushabh Parikh 22](#_Toc185172204)

[Stephanie Bravo-Heras 23](#_Toc185172205)

[Esther Bilenkin 23](#_Toc185172206)

[Appendix (Code) 24](#_Toc185172207)

[Programs for Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data 24](#_Toc185172208)

[Variance Based Program 24](#_Toc185172209)

[Correlation Matrix Program 26](#_Toc185172210)

[Elbow Program 28](#_Toc185172211)

[K-Means Program 30](#_Toc185172212)

[Random Forest Program 32](#_Toc185172213)

[DBSCAN 35](#_Toc185172214)

[Programs for Data Set 2: Bicycle Parking 37](#_Toc185172215)

[Variance Threshold 37](#_Toc185172216)

[Correlation Matrix Program 40](#_Toc185172217)

[Mean Shift 43](#_Toc185172218)

[Random Forest Program 44](#_Toc185172219)

# Introduction and Objectives

The project’s purpose is to explore the application of advanced data mining techniques to uncover patterns from the datasets used in Project 1. We chose the 2018 Central Park Squirrel Census dataset and the NYC Open Data Bicycle Parking dataset. The analysis done is meant to display the utility of clustering algorithms- K-Means, DBSCAN, and Mean Shift- as well as Random Forest classification to develop a deeper scope of insights.

The project’s objectives center on demonstrating the power of data-driven methods to address specific challenges in analytics. For the Squirrel Census dataset, the focus is on identifying behavioral trends of squirrels and the correlation of them with variables of time, such as the time of day. For the Bicycle Parking dataset, the focus is on identifying variables of space, such as geographical location, and analyzing the relationship between that and other attributes, like rack type, and external factors, like hurricane evacuation zones. By integrating clustering and classification methodologies, this project can highlight the strengths and weaknesses in dealing with real-world data, and how the results could inform further research and real-world decision-making, in this case, particularly in urban wildlife management and infrastructure planning.

# Description of Data Sets

## Data Set 1: [2018 Central Park Squirrel Census - Squirrel Data](https://data.cityofnewyork.us/Environment/2018-Central-Park-Squirrel-Census-Squirrel-Data/vfnx-vebw/about_data)

The dataset is a squirrel census conducted in New York City in 2018, containing 31 attributes and 3,024 records. It includes data such as the longitude, latitude, and date of each squirrel sighting, as well as various attributes describing the squirrels' characteristics, the type of area where they were observed, and the method of observation. Some of the data is sparse, particularly regarding the squirrels' behaviors (e.g., running, climbing, chasing, eating, foraging). There are also missing values related to squirrel coloration, which may require omitting incomplete rows or extrapolating the missing data.

Additionally, the column indicating whether the squirrel was seen above ground is inconsistent: if the squirrel was observed at ground level, the value is recorded as "False," but if observed above ground, the approximate height is recorded. This column will need to be standardized for consistency. There's also potential to explore the usability of the "Other Interactions" column, which appears to primarily record data about human-squirrel interactions.

Furthermore, we may consider transforming the categorical and binary data in the columns labeled "Kuks," "Quaas," "Moans," "Tail flags," "Tail twitches," and "Approaches," as some of these actions might be mutually exclusive and could be combined to streamline the data.

A white sheet with black text

Description automatically generated

Figure 1 A few records (rows) of the first dataset

## Data Set 2: [Bicycle Parking](https://data.cityofnewyork.us/Transportation/Bicycle-Parking/592z-n7dk/about_data)

The dataset used for the project is Bicycle Parking, provided by NYC Open Data. It represents all bike parking locations in the 5 boroughs of NYC, as provided by the DoT. They have, additionally, provided a map of the objects given in the dataset, though it is a separate download. It has been public since 2019 and is updated on an annual basis. It has 25 columns which generally define exacting details for where the parking lot is, as well as certain defining features of the lot, such as what type of bike rack it uses, when it was installed, and a unique identifier for the rack. Presently, it contains over 32,000 records.A white background with black text

Description automatically generated

Figure 2 A few records (rows) of the second dataset

# Design and Methods

### Approach

#### Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data

To understand how squirrel behavior varies by time of day (AM/PM), we took a comprehensive approach that involved data cleaning, feature selection, clustering, and predictive modeling. First, we cleaned the dataset to ensure its consistency and reliability, standardizing behavioral attributes, resolving redundancies, and removing noisy or contradictory data. Next, we performed feature selection, focusing on key behavioral traits—such as running, climbing, eating, and foraging—that showed sufficient variance and relevance for analysis. We then applied K-means and DBSCAN clustering techniques to identify patterns in squirrel activities. These methods revealed both common and rare behaviors that were previously unclear. Clustering uncovered time-specific trends, with certain clusters like 9, 22, and 6 showing a strong association with PM behaviors, while others, such as Cluster 0, were primarily AM. This clustering analysis provided valuable insights that served as the foundation for applying a Random Forest model to predict the time of day based on observed squirrel behaviors, allowing for a deeper exploration of the relationship between activity patterns and time.

#### Data Set 2: Bicycle Parking

Our approach for analyzing the Bicycle Parking data set is to apply both Mean Shift clustering and Random Forest classification. We want to clean up the data set by applying one-hot encoding to categorical variables, handling missing values, and performing attribute selection using variance thresholds to focus on the most relevant attributes. Our main goal is to identify patterns in the data set.

We want to focus on using the Mean Shift clustering algorithm to identify natural groupings within the bicycle parking dataset. We want to do this so that we can get insights into patterns, such as spatial clustering of parking racks or relationships between features like location, borough code, and rack type. By doing this we want to be able to use the unsupervised learning technique to group coordinates without predefined labels so that we can then observe how different attributes can cluster together and make patterns.

As a bonus, if we have time, we want to apply Random Forest classification to showcase the relationship between a borough and different attributes. This will allow us to apply prediction to our data set to evaluate each feature.

### Data Cleaning

#### Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data

The dataset was cleaned to resolve inconsistencies, standardize data, and reduce dimensionality. For the "Above Ground Sighter Measurement," ground-level sightings marked as "False" were changed to 0 to ensure all values were numeric. Blank values in the "Location" field, found alongside missing entries in "Above Ground Sighter Measurement," were inferred as "Ground Plane," updating 65 rows. Behavioral attributes—"Runs from," "Indifferent," and "Approaches"—were logically reconciled by removing contradictory records where more than one behavior was marked True. Cases where all three attributes were False were clarified using descriptions from the "Other Interactions" column, which was subsequently removed.

Dimensional reduction was achieved by eliminating non-essential attributes such as "Other Interactions," "Other Activities," and "Unique Squirrel ID," which added little analytical value. Attributes related to fur coloration, including "Primary Fur Color," "Highlight Fur Color," and "Combination of Primary and Highlight Color," were reviewed for missing values and renamed for clarity; the latter was updated to "Additionally Highlight Color" for accuracy. These changes ensured a cleaner, more consistent dataset better suited for analysis.

#### Data Set 2: Bicycle Parking

We needed to take data cleaning steps for our data because a lot of the data was noisy and had missing values. We applied one-hot encoding to the data set to update the categorical values to numeric values. When we did this, new columns were added in Boolean values, so we then proceeded to update the Boolean data into zeros or ones. We also got rid of attributes that were correlated to location like address, city, or borough name because they were redundant since we already had longitude and latitude. We also got rid of unique identifiers since they do not give us perspective into patterns that can be identified and are unique to each record. These might skew our data and are not recommended to keep.

### Attribute Selection

#### Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data

Using a Variance-Based Feature Selection algorithm, we found that certain attributes had enough variation to be useful for further analysis. This makes sense when we think about squirrel behavior. For example, if a squirrel is running, it’s probably not eating, interacting with humans, or twitching its tail. Similarly, if it’s eating, it’s less likely to be climbing or foraging. These patterns show why these attributes are important to study further.

* Running
* Climbing
* Eating
* Foraging
* TailTwitches
* HumanInteractions

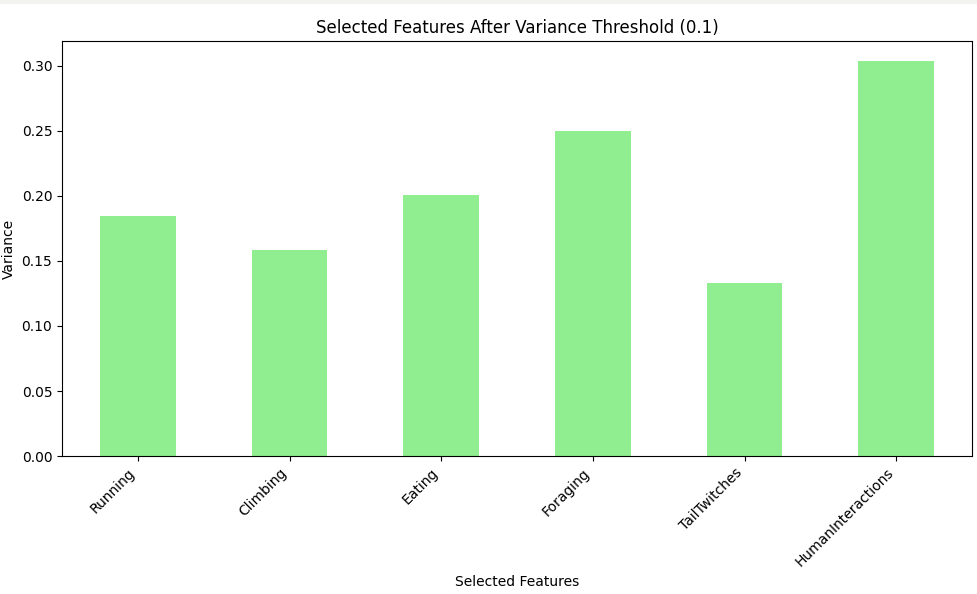


Figure 3 Attributes with Variance greater than 0.1 for Dataset 1

To reduce redundancy and understand the relationships between our attributes, we applied Correlation Matrix-Based Filtering. As expected, we observed a negative correlation between Eating and Foraging, since these behaviors are unlikely to occur simultaneously. Additionally, there was some correlation between Climbing and Kuks, and a strong correlation between Kuks and Quaas. Given the high correlation between Kuks and Quaas, we will only use one of these for k-means clustering to avoid redundancy. Additionally, Moans appear to be largely uncorrelated with other attributes and may introduce noise, so we will be excluded for better clustering results.

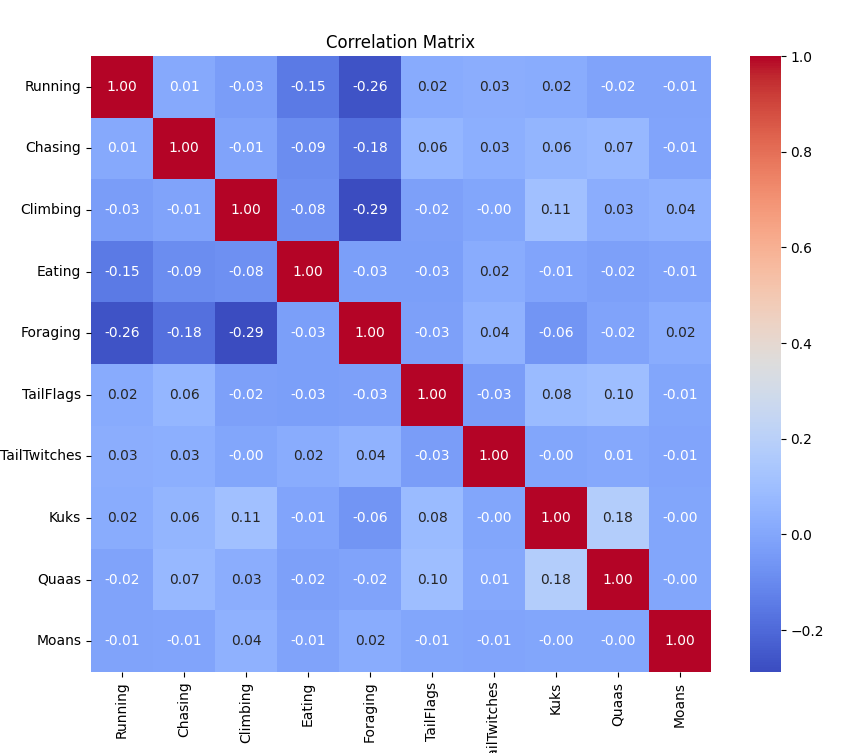


Figure 4 Correlation Matrix for Dataset 1

#### Data Set 2: Bicycle Parking

For Data Set Two, we also used the Variance Threshold algorithm to select the features that had enough variation to use when further analysis is applied. Any features that had a low variance are removed back they aren’t helpful to predicting behavior. For this data, set we looked for when there was a variance greater than 0.1 and therefore any feature below that threshold was removed unless they were our target variables (Longitude and Latitude). The RackType\_Large Hoop, RackType\_Small Hoop, and RackType\_U Rack features could be helpful because they have significant variability across the data which could be due to the diversity of rack types that are used in different locations. HrcEvac has enough variability that it could how evacuation zones might influence rack type. The BoroCode feature represents the different boroughs in Manhattan, and it can provide analytical value when looking at the impact patterns for rack types.

A graph with green bars

Description automatically generated

Figure 5 Attributes with Variance greater than 0.1 for Dataset 2

To further understand our data, we apply a correlation matrix which will show us the relationships between numerical attributes in our dataset. It will help us see data that is strongly correlated to each other, which can mean that the attribute carries redundant information. When attributes are highly correlated, it might negatively impact the clustering process. Reducing these attributes can make clusters more meaningful. When we look at the Correlation Matrix, it makes sense that the different rack type columns and FMAFIdz columns have negative correlation because they were originally one column each that got separated after applying One-Hot Encoding to the categorical attributes to make them numerical. We see a correlation that isn’t too positive or too negative between HrcEvac and the following attributes: RackType\_Small Hoop and RackType\_U Rack. This can signify that the evacuation zone might affect the rack type that is commonly used in an area. This confirms that the results from the Variance Thresholds could be used when applying it to our data training exercise. A screenshot of a graph

Description automatically generated

Figure 6 Correlation Matrix for Dataset 2

### Data Training

#### Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data

To determine the number of clusters for K-means clustering we first applied the elbow method. We discovered it was appropriate to use 7 clusters for the clustering.

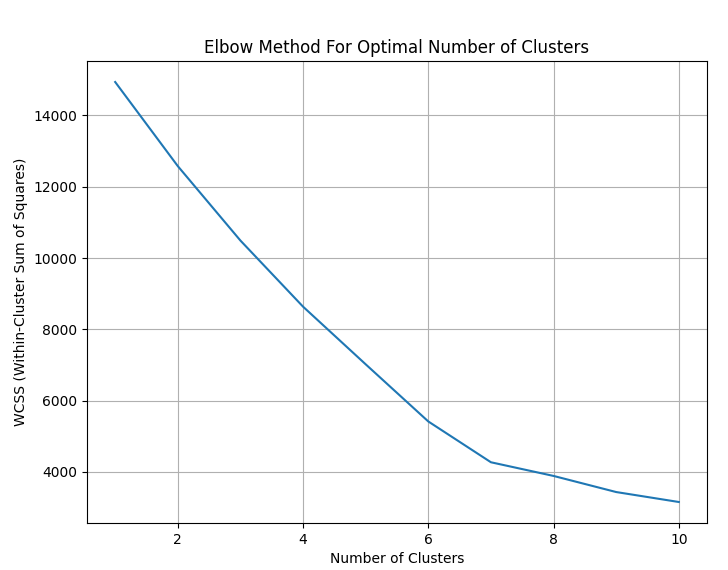


Figure 7 Correlation Matrix for Dataset 1

Using K-means clustering, we identified patterns in squirrel behavior and group sizes. Larger clusters, like Cluster 1 and Cluster 4, show common behaviors such as foraging and feeding, while smaller clusters, like Cluster 0 and Cluster 3, capture rare or unique behaviors like human interactions. Active behaviors like running, climbing, and vocalizing (Kuks) are more prominent in clusters like Cluster 6 and Cluster 1, whereas less active patterns appear in Cluster 0 and Cluster 3. Cluster 4 is notable for its strong focus on feeding, and some clusters emphasize vocalizations (Kuks), suggesting unique communication patterns among specific groups.

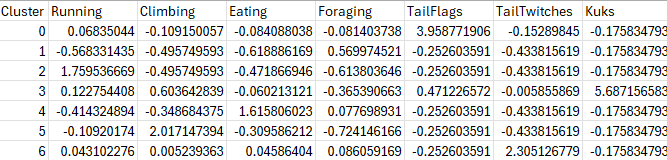


Figure 8 K-means cluster centers for Dataset 1



Figure 9.2 K-means cluster distribution for Dataset 1

Using Density-Based Spatial Clustering of Applications with Noise (DBSCAN), we observed that certain clusters are much larger than others, indicating that some behaviors are more common. Cluster 2, the largest, contains 518 samples, representing a dense group of squirrels with similar behaviors. In contrast, smaller clusters like 23 and 14, with only 10 samples each, highlight rare or less common behavior patterns. This distribution suggests that we can focus on larger clusters to study dominant traits, while smaller clusters and noise points may reveal unique or rare behaviors worth closer examination.

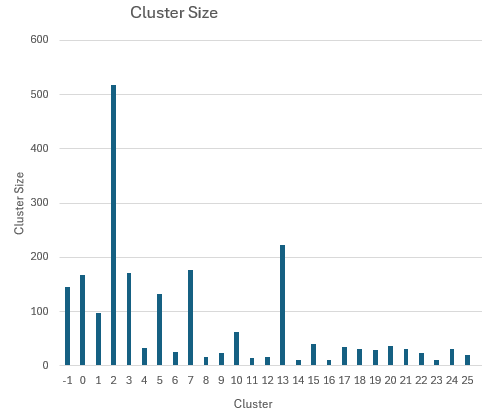


Figure 10 DBSCAN\_cluster distribution for Dataset 1

#### Data Set 2: Bicycle Parking

After preprocessing this data set and applying the variance threshold, we were able to select the attributes that would be used for Means Shift Clustering. The first step we took was standardizing all the attributes to make sure that every variable contributed equally. After we started clustering with the Mean Shift algorithm with a bandwidth of 2. This algorithm intends to identify clusters by finding dense regions in different locations. Each record was assigned to one of the nine clusters. The results are shown in the Cluster Map, where points with the same color represent them belonging to the same cluster. Most importantly, areas where clusters appear to have a denser grouping indicate attributes for bicycle parking that are similar or geographically close.

A screenshot of a table

Description automatically generated

Figure 11 Mean Shift Cluster Data Set

A blue and yellow cluster results

Description automatically generated

Figure 12 Mean Shift Cluster Map

# Results

## Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data

We aimed to explore whether the time of day (AM/PM) influenced squirrel behavior. To do this, we used a Random Forest model to analyze if behaviors such as running, chasing, climbing, eating, and foraging could predict whether the squirrel was observed in the morning or evening. While the model performed reasonably well at predicting morning behavior, it struggled to accurately predict nighttime behavior. After applying K-means and DBSCAN clustering, we found that some squirrel behaviors were overwhelmingly common, while others were much rarer. This distinction wasn't clear before clustering but incorporating the clustering results provided us with deeper insights into the patterns of squirrel behavior across different times of day.

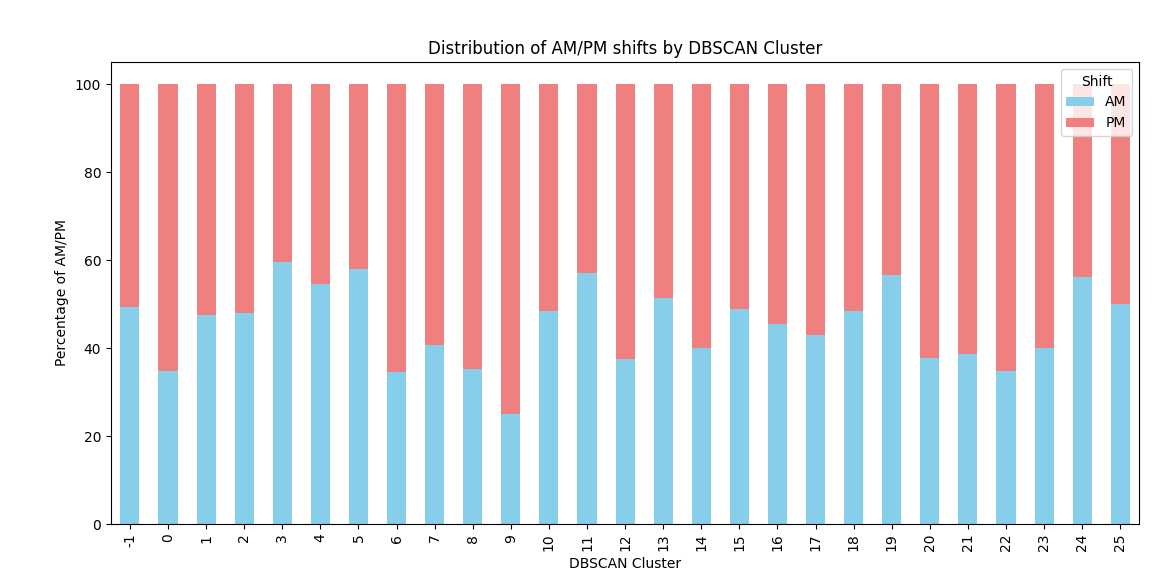


Figure 13 Random Forest for Dataset 1

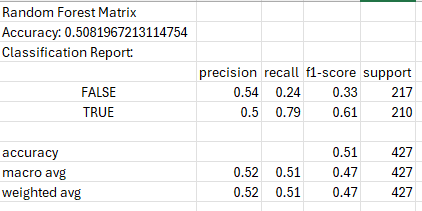


Figure 14 Random Forest Matrix for Dataset 1

Our analysis revealed that some clusters were strongly associated with either AM or PM, while others showed a more balanced distribution. For example, Clusters 9, 22, and 6 were predominantly PM, while Cluster 0 was mostly AM. This suggests that the behaviors in these clusters may be linked to specific times of day. With this information, we can refine our training models and focus on clusters like 9, 22, and 6 to further investigate squirrel behavior at night or on 0 to study behaviors in the morning.

## Data Set 2: Bicycle Parking

Our goal for this data set was to be able to see if there are patterns in bicycle parking when it comes to neighborhoods. This was the main reason we chose the Mean Shift Clustering method for this data set instead of the ones used in data set one. After applying this method, we were able to confirm that there were distinct groupings in the bicycle parking data based on the different attributes. This can help real-world situations understand patterns in bicycle parking and allow different entities to be able to make informed decisions when it comes to improving bicycle parking infrastructure. We decided to take our analysis a step further and apply the Random Forest Classification to our data set as well to predict the likelihood of certain bike racks being used in specific locations and identify the most influential attributes affecting these outcomes. The Random Forest classification is predicting the BoroCode for each sample in the data set which we know represents different NYC boroughs. We removed longitude and latitude from the data set for this because the accuracy when we kept them was close to 1. We understood that the reason why is because the coordinates were highly correlated to the BoroCode. After we dropped those attributes, the accuracy dropped drastically to 0.4825, which made more sense to us. The borough 1 (Manhattan) model performed fairly as it had a precision of 0.48, a Recall of 0.49, and an F-1-Score of 0.49. This means that it is correctly identifying about 49% of all its cases. Borough 2 (The Bronx) performed extremely poorly, having a 0.00 for all three measures, meaning that the data set provided no insights since it predicted nothing. Borough 3 (Brooklyn) performed okay with a Precision of 0.49, a Recall of 0.61, and an F-1 Score of 0.54. This means that although this borough is predicted more, many of the predictions are incorrect. Borough 4 (Queens) has a Precision of 0.47, a Recall of 0.56, and an F-1 Score of 0.51. Lastly, borough 5 (Staten Island) is poorly predicted with a Precision of 0.36, a Recall of 0.03, and an F-1 Score of 0.05. The overall accuracy of 0.48% indicates that the model only identifies the borough correctly 48% of the time. When looking at the Feature Importance, we can see that the top 4 attributes are the ones we used in our Mean Shift Clustering.

A graph with blue bars

Description automatically generated

Figure 15 Random Forest Feature Importance

A screenshot of a computer

Description automatically generated

Figure 16 Calculations for BoroCode

We also went ahead and changed our target value to the HrcEvac attribute and got an accuracy of 0.6368, which makes me think that it was easier to predict the hurricane evacuation zone than the borough using the test data. A screenshot of a computer screen

Description automatically generated

Figure 17 Calculations for BoroCode

# Conclusions and Future Works

## Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data

Our analysis demonstrated that clustering effectively revealed behavioral patterns linked to specific times of day, providing valuable insights that enhanced the performance of the Random Forest model. While the model performed well in predicting morning behaviors, it struggled with evening predictions due to the greater variability in PM activities. By focusing on clusters with strong time-specific associations, such as Clusters 9, 22, and 6 (PM) and Cluster 0 (AM), we can refine the model and improve its accuracy. These clusters offer a promising basis for further analysis of squirrel behavior, allowing us to investigate how certain behaviors, like foraging or human interactions, vary throughout the day.

Additionally, the time-specific clusters can inform new studies by targeting particular behaviors in the AM or PM periods. Researchers can design more focused studies that explore specific clusters in greater detail, examining how environmental factors, like food availability or human presence, influence squirrel activity at different times of day. This could lead to deeper insights into animal behavior and how it adapts to changing conditions, ultimately helping to shape more targeted wildlife conservation efforts and urban ecology studies.

## Data Set 2: Bicycle Parking

We were able to look at the relationships between different attributes in the data set and bicycle parking locations. Originally, we had believed that there would be a strong correlation between rack type, coordinates, and hurricane evacuation zones. However, the results of all the analyses reveal that there is a moderate level of correlation which could suggest different things. There could be lurking variables that are not represented in the data set, or it could be due to the data cleaning and attribute selection exercise that was done prior to applying the Mean Shift clustering and Random Forest classification. Another factor could be the algorithms chosen, perhaps there could have been another algorithm that would have worked better with the data set that we originally disposed of like the algorithms used in data set 1. In the future, we would like to be able to apply the algorithms not only used in data set 1 but also other algorithms we learned in class.

# Team Member’s Contributions

## Group Activities:

Everyone in the group looked at the different data sets from the first project and decided which of the three to use. Then they discussed which data mining task they believed would work best with both data sets. Once they decided on clustering, they also investigated which algorithms to use for each or both data sets. They also decided to apply Random Forest to showcase classification (another data mining task) for each data set.

## Rushabh Parikh

Rushabh worked on dataset 1, focusing on analyzing squirrel behavior in 2018. He was responsible for cleaning and preprocessing the data, followed by feature selection and model training using techniques like K-means, DBSCAN, and Random Forest. Rushabh implemented feature selection methods, including the Correlation Matrix and Variance-Based techniques, and wrote the necessary code for these steps. He also documented the approach, detailing the data cleaning, feature selection, model training, and analysis, and contributed to the results and conclusions of the project.

## Stephanie Bravo-Heras

Stephanie worked on setting up the Word document and outlining all the requirements needed to complete this project. She analyzed which data preprocessing activities she needed to take and performed data cleaning steps on the Bicycle Parking data set. She wrote code for the One-Hot Encoding steps to convert categorical attributes into numerical attributes. This ended up creating Boolean variables, so she also wrote code to turn the Boolean into numeric values. After, she wrote code to apply the Variance Threshold and Correlation Matrix for this data set before writing the code for the Mean Shift Cluster and the Random Forest classification that were used specific to her data set.

## Esther Bilenkin

Esther contributed to the project by helping select the datasets and providing input on the data mining tasks, including the choice of clustering algorithms (K-means, DBSCAN, and Mean Shift) and Random Forest classification. She assisted with refining the report and offering feedback to improve the clarity of sections on data cleaning, attribute selection, and results. She also reviewed and tested implementations, ensuring consistency across datasets and aligned with the objectives. She also collaborated in managing deadlines, proofreading the final document, and reviewing statistical outputs and visualizations.

# Appendix (Code)

## Programs for Data Set 1: 2018 Central Park Squirrel Census - Squirrel Data

### Variance Based Program

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.feature\_selection import VarianceThreshold

from sklearn.preprocessing import LabelEncoder

# Load the dataset

data = pd.read\_csv("2018\_Central\_Park\_Squirrel\_Census\_-\_Squirrel\_Data\_20241013 updated.csv")

# Select relevant features (excluding the target variable)

features = ['Running', 'Chasing', 'Climbing', 'Eating', 'Foraging', 'TailFlags', 'TailTwitches', 'Kuks', 'Quaas', 'HumanInteractions']

# Preprocess features

# Convert boolean-like columns (TailFlags, Kuks, Quaas, Moans, TailTwitches) to numeric (True=1, False=0)

data\_encoded = data[features].applymap(lambda x: 1 if x == "True" else (0 if x == "False" else x))

# Encode categorical variables (e.g., HumanInteractions) to numeric values

# Use LabelEncoder to encode 'HumanInteractions' (which contains string values like 'Runs From', 'Indifferent', etc.)

le = LabelEncoder()

data\_encoded['HumanInteractions'] = le.fit\_transform(data\_encoded['HumanInteractions'])

# Calculate the variance of each feature

feature\_variance = data\_encoded.var()

# Plot the variance of each feature before applying VarianceThreshold

plt.figure(figsize=(10, 6))

feature\_variance.plot(kind='bar', color='skyblue')

plt.title("Variance of Each Feature Before Variance Threshold")

plt.xlabel("Features")

plt.ylabel("Variance")

plt.xticks(rotation=45, ha="right")

plt.tight\_layout()

plt.show()

# Apply Variance Threshold with a threshold of 0.1

selector = VarianceThreshold(threshold=0.1)

data\_selected = selector.fit\_transform(data\_encoded)

# Get the selected features (those with variance > 0.1)

selected\_features = data\_encoded.columns[selector.get\_support()]

# Show selected features

print("Selected Features after Variance Threshold (0.1):", selected\_features)

# Visualize the selected features after applying VarianceThreshold

plt.figure(figsize=(10, 6))

selected\_variance = feature\_variance[selector.get\_support()]

selected\_variance.plot(kind='bar', color='lightgreen')

plt.title("Selected Features After Variance Threshold (0.1)")

plt.xlabel("Selected Features")

plt.ylabel("Variance")

plt.xticks(rotation=45, ha="right")

plt.tight\_layout()

plt.show()

### Correlation Matrix Program

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Load the dataset

data = pd.read\_csv("2018\_Central\_Park\_Squirrel\_Census\_-\_Squirrel\_Data\_20241013 updated.csv")

# Select relevant features

features = ['Running', 'Chasing', 'Climbing', 'Eating', 'Foraging', 'TailFlags', 'TailTwitches', 'Kuks', 'Quaas', 'Moans']

# Preprocess the data

# Convert the boolean-like columns (TailFlags, Kuks, Quaas, Moans, TailTwitches) to numeric (True=1, False=0)

# Using 'apply' instead of 'applymap' for compatibility

data\_encoded = data[features].apply(lambda x: pd.to\_numeric(x, errors='coerce'))

# Now, make sure all the data is numeric (if any errors were encountered, they will be NaN, which we can handle)

data\_encoded = data\_encoded.fillna(0) # Filling NaN with 0 for simplicity

# Compute the correlation matrix

correlation\_matrix = data\_encoded.corr()

# Visualize the correlation matrix using a heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Matrix')

plt.show()

correlation\_matrix.to\_csv("correlation\_matrix.csv")

### Elbow Program

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Load the dataset

data = pd.read\_csv("2018\_Central\_Park\_Squirrel\_Census\_-\_Squirrel\_Data\_20241013 updated.csv")

# Select the relevant features based on both variance-based and correlation matrix-based filtering

selected\_features = ['Running', 'Climbing', 'Eating', 'Foraging', 'TailFlags', 'TailTwitches', 'Kuks'] # Excluding Moans due to weak correlation

# Preprocess the data: Convert any boolean-like columns to numeric

data\_encoded = data[selected\_features].apply(lambda x: pd.to\_numeric(x, errors='coerce'))

# Handle missing values (NaN) by filling with 0 (for simplicity)

data\_encoded = data\_encoded.fillna(0)

# Standardize the data

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(data\_encoded)

# Apply the Elbow Method to find the optimal number of clusters

wcss = []

for i in range(1, 11): # Checking for cluster numbers from 1 to 10

kmeans = KMeans(n\_clusters=i, init='k-means++', max\_iter=300, n\_init=10, random\_state=42)

kmeans.fit(data\_scaled)

wcss.append(kmeans.inertia\_)

# Plotting the Elbow curve

plt.figure(figsize=(8, 6))

plt.plot(range(1, 11), wcss)

plt.title('Elbow Method For Optimal Number of Clusters')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS (Within-Cluster Sum of Squares)')

plt.grid(True)

plt.show()

### K-Means Program

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Load the dataset

data = pd.read\_csv("2018\_Central\_Park\_Squirrel\_Census\_-\_Squirrel\_Data\_20241013 updated.csv")

# Select the relevant features based on both variance-based and correlation matrix-based filtering

selected\_features = ['Running', 'Climbing', 'Eating', 'Foraging', 'TailFlags', 'TailTwitches', 'Kuks'] # Excluding Moans due to weak correlation

# Preprocess the data: Convert any boolean-like columns to numeric

data\_encoded = data[selected\_features].apply(lambda x: pd.to\_numeric(x, errors='coerce'))

# Handle missing values (NaN) by filling with 0 (for simplicity)

data\_encoded = data\_encoded.fillna(0)

# Standardize the data

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(data\_encoded)

# Apply K-Means Clustering with 7 clusters

kmeans = KMeans(n\_clusters=7, init='k-means++', max\_iter=300, n\_init=10, random\_state=42)

data['Cluster'] = kmeans.fit\_predict(data\_scaled)

# Show the cluster centers and the distribution of data in each cluster

print("Cluster Centers:")

print(kmeans.cluster\_centers\_)

# Show how many samples belong to each cluster

print("\nCluster Distribution:")

print(data['Cluster'].value\_counts())

# Export the data with clusters to a CSV file

data.to\_csv("squirrel\_data\_with\_clusters.csv", index=False)

# Visualize the clusters

plt.figure(figsize=(8, 6))

plt.scatter(data\_scaled[:, 0], data\_scaled[:, 1], c=data['Cluster'], cmap='rainbow', s=50)

plt.title('Clusters of Squirrel Data (First Two Features)')

plt.xlabel('Feature 1: ' + selected\_features[0])

plt.ylabel('Feature 2: ' + selected\_features[1])

plt.colorbar(label='Cluster')

plt.show()

### Random Forest Program

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

# Load the dataset

data = pd.read\_csv("2018\_Central\_Park\_Squirrel\_Census\_-\_Squirrel\_Data\_20241013\_updated\_dbscan\_clusters.csv")

# Select relevant features and target variable

features = ['Running', 'Chasing', 'Climbing', 'Eating', 'Foraging', 'DBSCAN\_Cluster'] # Added DBSCAN\_Cluster

target = 'Shift' # Target variable (AM/PM)

# Preprocess the data

# Encode the target (Shift) using LabelEncoder (since it's categorical)

le = LabelEncoder()

target\_encoded = le.fit\_transform(data[target]) # Encode 'Shift' to numeric values (AM=0, PM=1)

# Preprocess features

# Convert the boolean-like columns to numeric (True=1, False=0)

data\_encoded = data[features].applymap(lambda x: 1 if x == "True" else (0 if x == "False" else x))

# Ensure that DBSCAN\_Cluster is treated as a numerical feature

data\_encoded['DBSCAN\_Cluster'] = pd.to\_numeric(data['DBSCAN\_Cluster'], errors='coerce') # Convert to numeric, handling any non-numeric values

# Split into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data\_encoded, target\_encoded, test\_size=0.2, random\_state=42)

# Train a Random Forest Classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X\_train, y\_train)

# Evaluate the model

y\_pred = clf.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# Visualize the feature importances

plt.figure(figsize=(12, 6))

plt.barh(data\_encoded.columns, clf.feature\_importances\_, color='skyblue')

plt.xlabel("Feature Importance")

plt.ylabel("Features")

plt.title("Feature Importance for Predicting Shift (AM/PM)")

plt.show()

# DBSCAN\_Cluster analysis for Shift distribution

shift\_distribution = data.groupby(['DBSCAN\_Cluster', 'Shift']).size().unstack(fill\_value=0)

# Normalize the distribution to get the percentage

shift\_distribution\_percentage = shift\_distribution.div(shift\_distribution.sum(axis=1), axis=0) \* 100

# Print the distribution

print("\nShift Distribution by DBSCAN Cluster (Percentage):")

print(shift\_distribution\_percentage)

# Plot the distribution for each cluster

shift\_distribution\_percentage.plot(kind='bar', stacked=True, figsize=(10, 6), color=['skyblue', 'lightcoral'])

plt.title('Distribution of AM/PM shifts by DBSCAN Cluster')

plt.xlabel('DBSCAN Cluster')

plt.ylabel('Percentage of AM/PM')

plt.xticks(rotation=90)

plt.legend(title='Shift', labels=['AM', 'PM'])

plt.show()

### DBSCAN

import pandas as pd

from sklearn.cluster import DBSCAN

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

# Load the dataset

data = pd.read\_csv("2018\_Central\_Park\_Squirrel\_Census\_-\_Squirrel\_Data\_20241013 updated.csv")

# Select relevant features

selected\_features = ['Running', 'Climbing', 'Eating', 'Foraging', 'TailFlags', 'TailTwitches', 'Kuks']

# Preprocess the data: Convert any boolean-like columns to numeric

data\_encoded = data[selected\_features].apply(lambda x: pd.to\_numeric(x, errors='coerce')).fillna(0)

# Standardize the data

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(data\_encoded)

# Apply DBSCAN

dbscan = DBSCAN(eps=0.5, min\_samples=10) # Adjust `eps` and `min\_samples` for optimal clustering

data['DBSCAN\_Cluster'] = dbscan.fit\_predict(data\_scaled)

# Show the number of points in each cluster

cluster\_counts = data['DBSCAN\_Cluster'].value\_counts()

print("\nCluster Distribution:")

print(cluster\_counts)

# Export data with clusters

data.to\_csv("squirrel\_data\_with\_dbscan\_clusters.csv", index=False)

# Visualize the clusters (using the first two features for simplicity)

plt.figure(figsize=(8, 6))

plt.scatter(data\_scaled[:, 0], data\_scaled[:, 1], c=data['DBSCAN\_Cluster'], cmap='rainbow', s=50)

plt.title('DBSCAN Clusters of Squirrel Data (First Two Features)')

plt.xlabel('Feature 1: ' + selected\_features[0])

plt.ylabel('Feature 2: ' + selected\_features[1])

plt.colorbar(label='Cluster')

plt.show()

## Programs for Data Set 2: Bicycle Parking

### Variance Threshold

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.feature\_selection import VarianceThreshold

import numpy as np

# Load the dataset

df = pd.read\_csv("Bicycle\_Parking\_20241213\_Updated.csv")

# Check for missing values and drop them

print(df.isnull().sum())

df = df.dropna()

#Get non-numeric values

non\_numeric\_columns = df.select\_dtypes(exclude=np.number).columns

print("The following columns are non-numeric")

print(non\_numeric\_columns)

#Index(['Program', 'RackType', 'FEMAFldz'], dtype='object')

# Perform One-Hot Encoding on non-numeric columns

df\_encoded = pd.get\_dummies(df, columns=non\_numeric\_columns, drop\_first=True)

print("Data after One-Hot Encoding:")

print(df\_encoded.head())

# Inspect encoded data

print("Encoded DataFrame Info:")

print(df\_encoded.info())

# Identify boolean columns (created by One-Hot Encoding)

bool\_columns = df\_encoded.select\_dtypes(include=bool).columns

print("Boolean Columns Found After One-Hot Encoding:", bool\_columns)

# Convert boolean columns to numeric (True = 1, False = 0)

df\_encoded[bool\_columns] = df\_encoded[bool\_columns].astype(int)

# Confirm conversion

print("After Converting Boolean Columns:")

print(df\_encoded[bool\_columns].head())

# Calculate the variance of each feature

feature\_variance = df\_encoded.var()

print("Feature Variances:")

print(feature\_variance)

# Plot the variance of each feature before applying VarianceThreshold

plt.figure(figsize=(10, 6))

feature\_variance.plot(kind='bar', color='skyblue')

plt.title("Variance of Each Feature Before Variance Threshold")

plt.xlabel("Features")

plt.ylabel("Variance")

plt.xticks(rotation=45, ha="right", fontsize=8)

plt.tight\_layout()

plt.show()

# Apply Variance Threshold with a threshold of 0.1

selector = VarianceThreshold(threshold=0.1)

df\_selected = selector.fit\_transform(df\_encoded)

# Get the selected features (those with variance > 0.1)

selected\_features = df\_encoded.columns[selector.get\_support()]

# Show selected features

print("Selected Features after Variance Threshold (0.1):", selected\_features)

# Visualize the selected features after applying VarianceThreshold

plt.figure(figsize=(10, 6))

selected\_variance = feature\_variance[selector.get\_support()]

selected\_variance.plot(kind='bar', color='lightgreen')

plt.title("Selected Features After Variance Threshold (0.1)")

plt.xlabel("Selected Features")

plt.ylabel("Variance")

plt.xticks(rotation=45, ha="right", fontsize=8)

plt.yticks(np.arange(0, step=0.2),rotation=45, ha="right", fontsize=8)

plt.tight\_layout()

plt.show()

### Correlation Matrix Program

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

# Load the dataset

df = pd.read\_csv("Bicycle\_Parking\_20241213\_Updated.csv")

# Check for missing values and drop them

print(df.isnull().sum())

df = df.dropna()

#Get non-numeric values

non\_numeric\_columns = df.select\_dtypes(exclude=np.number).columns

print("The following columns are non-numeric")

print(non\_numeric\_columns)

#Index(['Program', 'RackType', 'FEMAFldz'], dtype='object')

# Perform One-Hot Encoding on non-numeric columns

df\_encoded = pd.get\_dummies(df, columns=non\_numeric\_columns, drop\_first=True)

print("Data after One-Hot Encoding:")

print(df\_encoded.head())

# Inspect encoded data

print("Encoded DataFrame Info:")

print(df\_encoded.info())

# Identify boolean columns (created by One-Hot Encoding)

bool\_columns = df\_encoded.select\_dtypes(include=bool).columns

print("Boolean Columns Found After One-Hot Encoding:", bool\_columns)

# Convert boolean columns to numeric (True = 1, False = 0)

df\_encoded[bool\_columns] = df\_encoded[bool\_columns].astype(int)

# Confirm conversion

print("After Converting Boolean Columns:")

print(df\_encoded[bool\_columns].head())

# Compute the correlation matrix

correlation\_matrix = df\_encoded.corr()

# Visualize the correlation matrix using a heatmap

plt.figure(figsize=(100, 80))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Matrix')

plt.show()

correlation\_matrix.to\_csv("correlation\_matrix.csv")

### Mean Shift

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import MeanShift

from sklearn.preprocessing import StandardScaler

# Load your dataset

df = pd.read\_csv("Bicycle\_Parking\_20241213\_AfterConversion.csv")

# Relevant features for clustering based on Variance Threshold

features = ['Longitude','Latitude','BoroCode', "HrcEvac",'RackType\_Large Hoop','RackType\_U Rack','RackType\_Small Hoop']

# Standardize the features

scaler = StandardScaler()

df\_scaled = scaler.fit\_transform(df[features])

#Mean Shift clustering Algorithm

mean\_shift = MeanShift(bandwidth=2)

df['Cluster'] = mean\_shift.fit\_predict(df\_scaled)

df.to\_csv("bicycle\_data\_with\_clusters.csv", index=False)

# Visualizing the clustering result

plt.figure(figsize=(8, 6))

plt.scatter(df['Longitude'], df['Latitude'], c=df['Cluster'], cmap='cividis', marker='o')

plt.title("Mean Shift Cluster Results")

plt.xlabel('Longitude')

plt.ylabel('Latitude')

plt.colorbar(label='Cluster')

plt.show()

# Display the number of clusters

num\_clusters = len(set(df['Cluster']))

print(f"Number of clusters identified: {num\_clusters}")

### Random Forest Program

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.preprocessing import LabelEncoder

# Load dataset

df = pd.read\_csv("Bicycle\_Parking\_20241213\_AfterConversion.csv")

# Target variable

target = 'BoroCode'

X = df.drop(columns=[target])

y = df[target]

# 80% train data , 20% test data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# StartRandom Forest Classifier

classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

# Train the model on the training data

classifier.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = classifier.predict(X\_test)

# Evaluate the model's performance

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.4f}")

# Display classification report (precision, recall, F1-score)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

# Feature Importance

importances = classifier.feature\_importances\_

cols = X.columns

# Sort feature importances in descending order

sorted\_cols = importances.argsort()[::-1]

# Print the feature importance

print("\nFeature Importance:")

for i in sorted\_cols:

print(f"{cols[i]}: {importances[i]:.4f}")

# Visualize Rand Forest Results

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))

plt.barh(cols[sorted\_cols], importances[sorted\_cols], color='skyblue')

plt.xlabel('Feature Importance')

plt.title('Random Forest Results')

plt.tight\_layout()

plt.show()