Data Literacy

University of Tübingen, Winter Term 2021/22

Analysis and Visualization of 2018 Central Park Squirrel Census - Squirrel Data

In 2018 there has been a census of the population of squirrels in the New York Central Park. In this paper, we analyze and visualize this publicly available data to investigate patterns in squirrel color and behaviour.

For multiple years now, a census has been made of the New York Central Park squirrel population. The data is collected with the help of volunteers, who name themselves Squirrel Sighters and made available to the public on their webpage. In this paper we will work with the data from the 2018 Central Park Squirrel Census. It was collected by 323 volunteers and comprises a total of 3023 squirrel sightings. These do not necessarily represent unique squirrels and the official estimate is instead 2373, for a more precise estimate however, tagging would likely be required. For our project, we have chosen to look into a variety of distributions potentially present in the data and visualized a selection of them.

Importing Packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.pyplot import cm
from matplotlib.axes._axes import _log as matplotlib_axes_logger
matplotlib_axes_logger.setLevel('ERROR')
!pip install tueplots
from tueplots import bundles
plt.rcParams.update(bundles.neurips2021(usetex=False))
```

```
Requirement already satisfied: tueplots in
/usr/local/lib/python3.7/dist-packages (0.0.3)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.7/dist-packages (from tueplots) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-
packages (from tueplots) (1.19.5)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->tueplots)
(1.3.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!
=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from
matplotlib->tueplots) (3.0.7)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->tueplots)
(0.11.0)
```

```
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->tueplots)
(2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1-
>matplotlib->tueplots) (1.15.0)
```

```
from src.utils import *
from src.make_plots import *
from src.pred_utils import *
from src.condProb import *
```

Data deployment

```
url =
"https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/
data/2019/2019-10-29/nyc_squirrels.csv"
nyc_squirrels = pd.read_csv(url)
nyc_squirrels.rename(columns={'shift': 'slot'}, inplace=True)
nyc_squirrels.head()
```

```
        long        lat  ... city_council_districts police_precincts
0 -73.956134  40.794082  ...                     19               13
1 -73.957044  40.794851  ...                     19               13
2 -73.976831  40.766718  ...                     19               13
3 -73.975725  40.769703  ...                     19               13
4 -73.959313  40.797533  ...                     19               13

[5 rows x 36 columns]
```

Data cleaning

Dealing with unwanted entries, dropping the unwanted columns

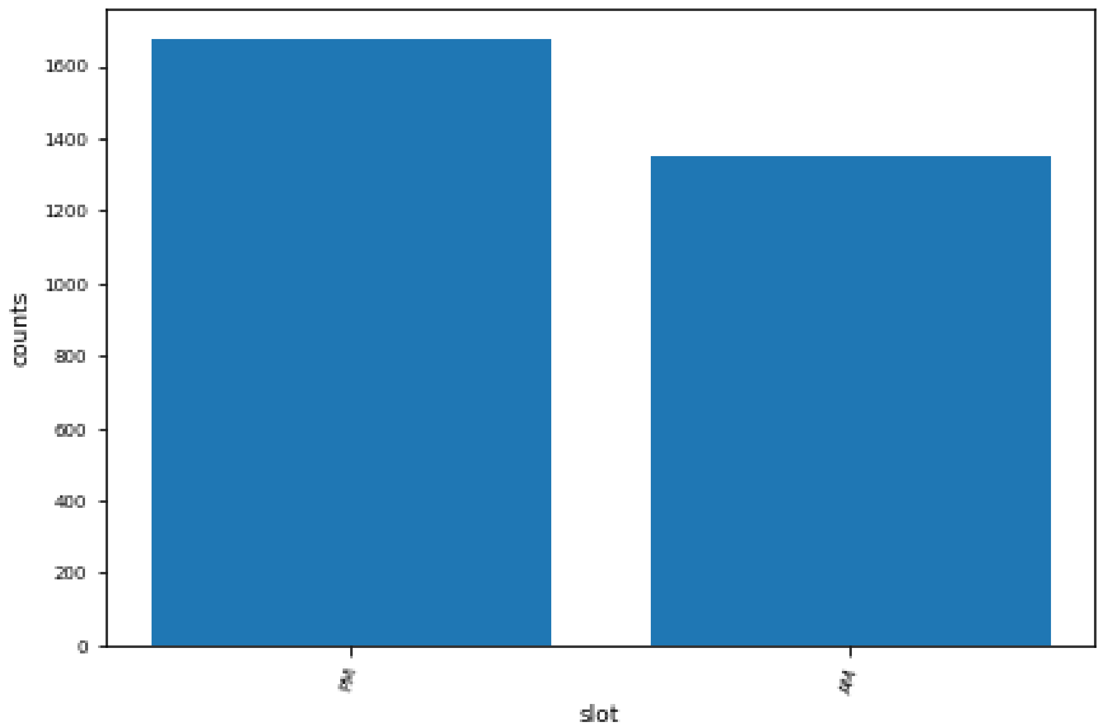
```
nyc_squirrels = dataset_fix(nyc_squirrels)
```

Counting plots

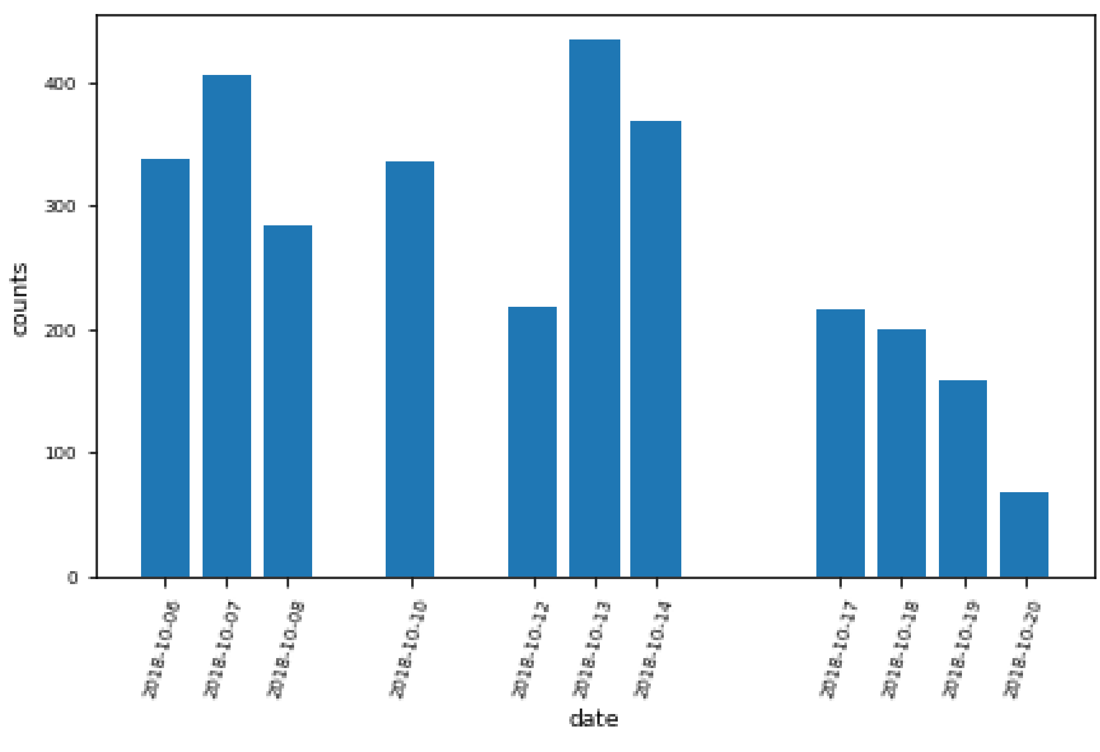
Counting individual features to see what is in the data

```
plot_counts(nyc_squirrels)
interest_list = ['slot', 'date', 'age', 'primary_fur_color',
       'highlight_fur_color', 'running', 'chasing', 'climbing',
'eating',
       'foraging', 'kuks', 'quaas', 'moans', 'tail_flags',
       'tail_twitches', 'approaches', 'indifferent', 'runs_from']
for c in interest_list:
  plot_counts(nyc_squirrels, plotted_feature=c, save_fig=False)
```

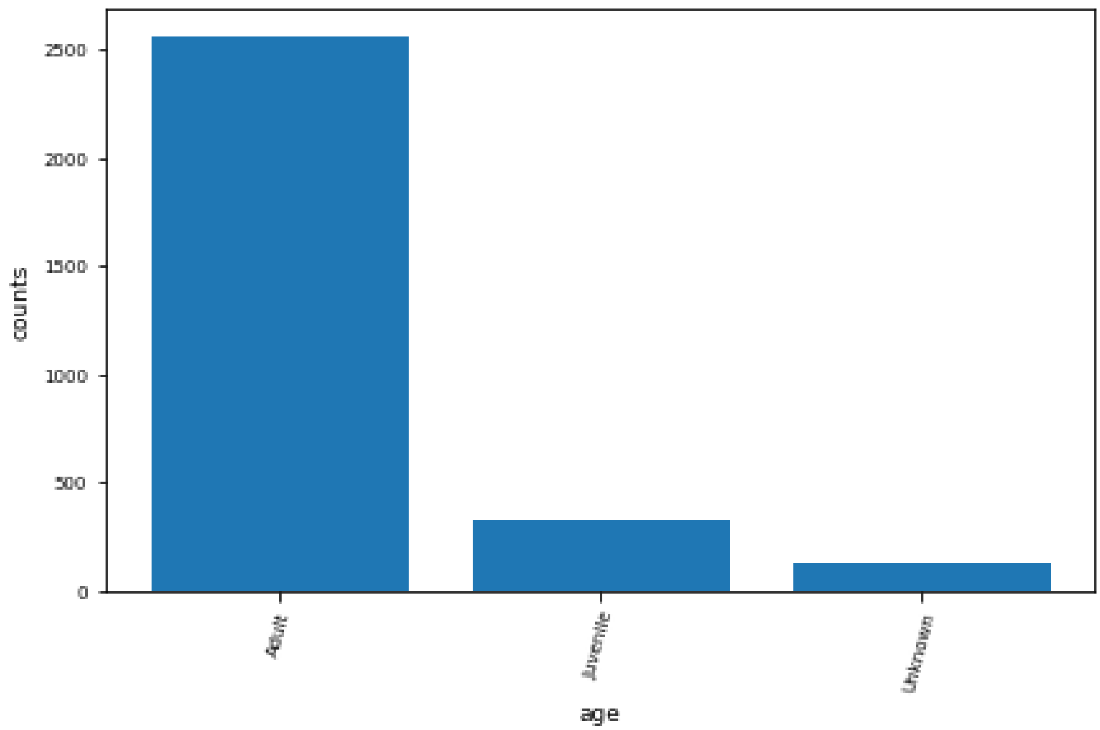
```
Number of observations is 3023.
```



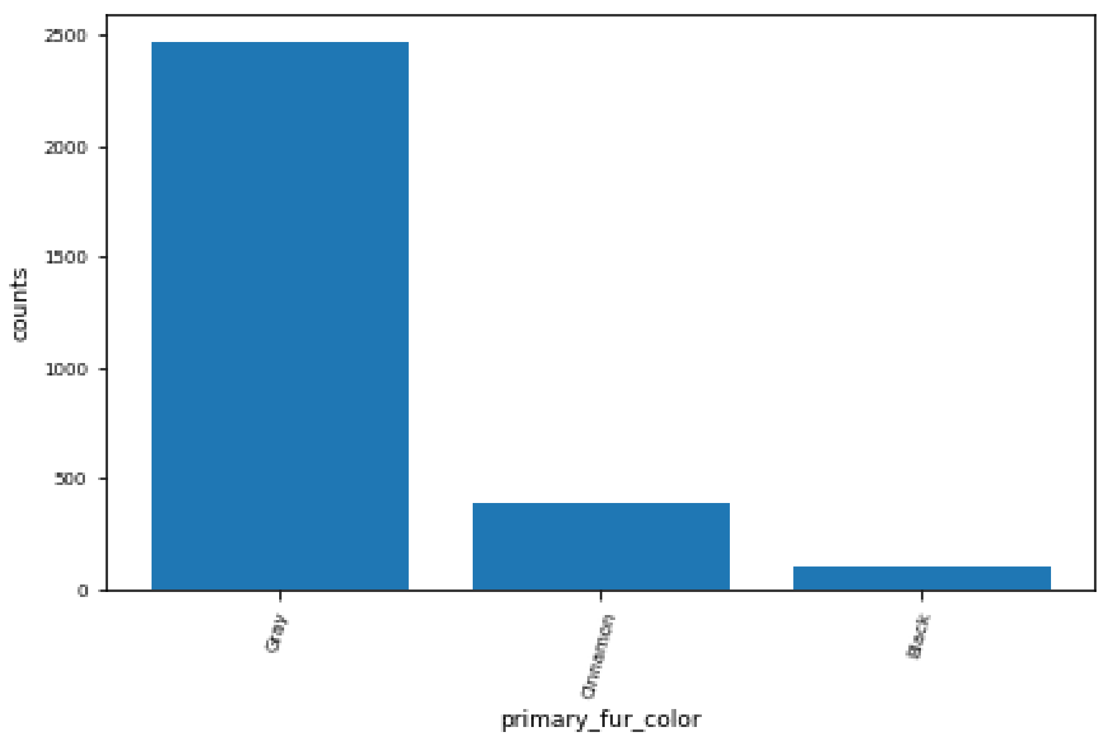
counts
1600
1400
1200
1000
800
600
400
200
0
PM
AM
slot



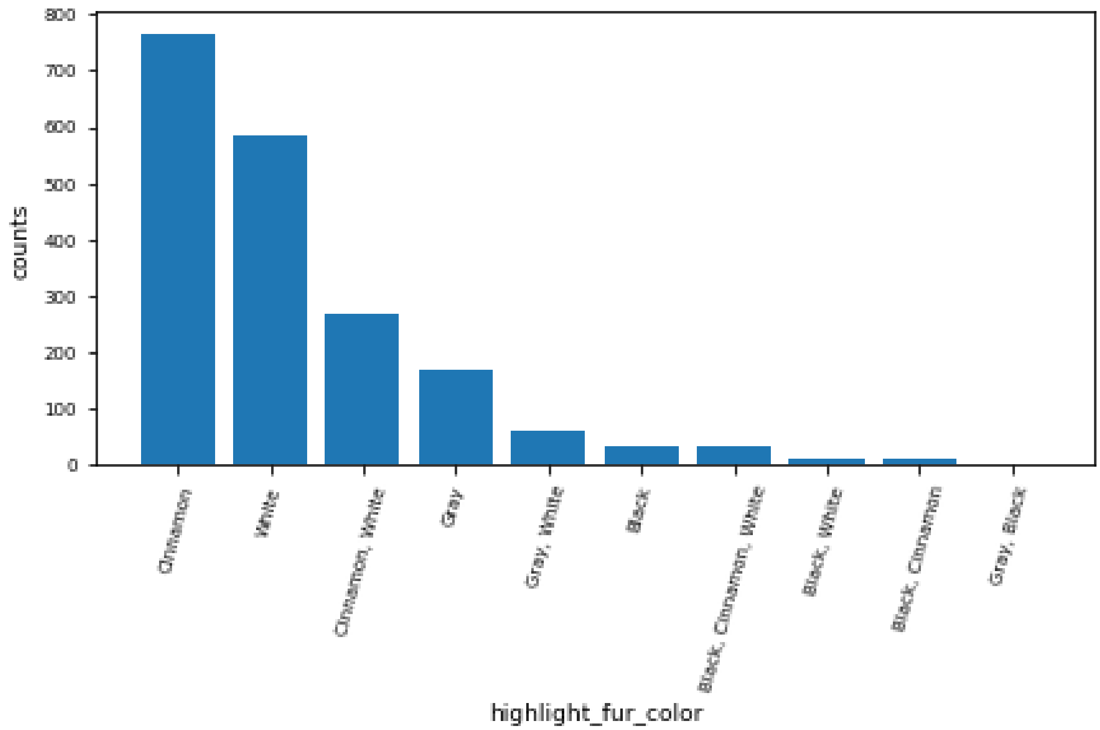
counts
400
300
200
100
0
2018-10-06
2018-10-07
2018-10-08
2018-10-10
2018-10-12
2018-10-13
2018-10-14
2018-10-17
2018-10-18
2018-10-19
2018-10-20
date

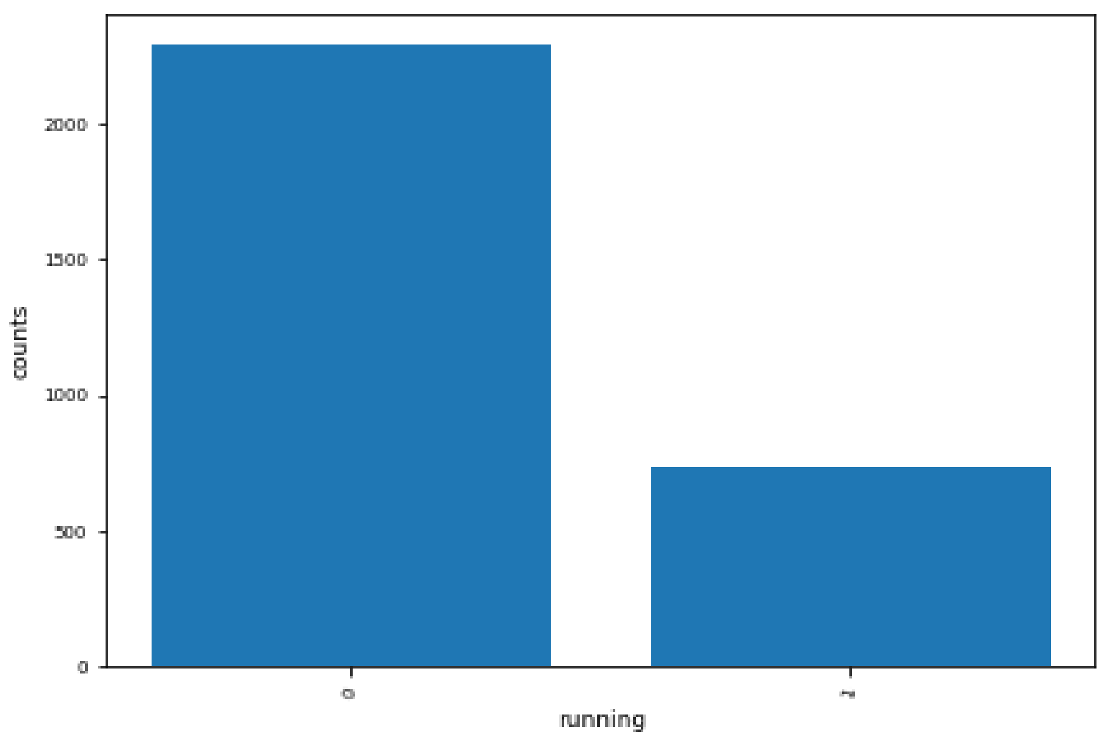


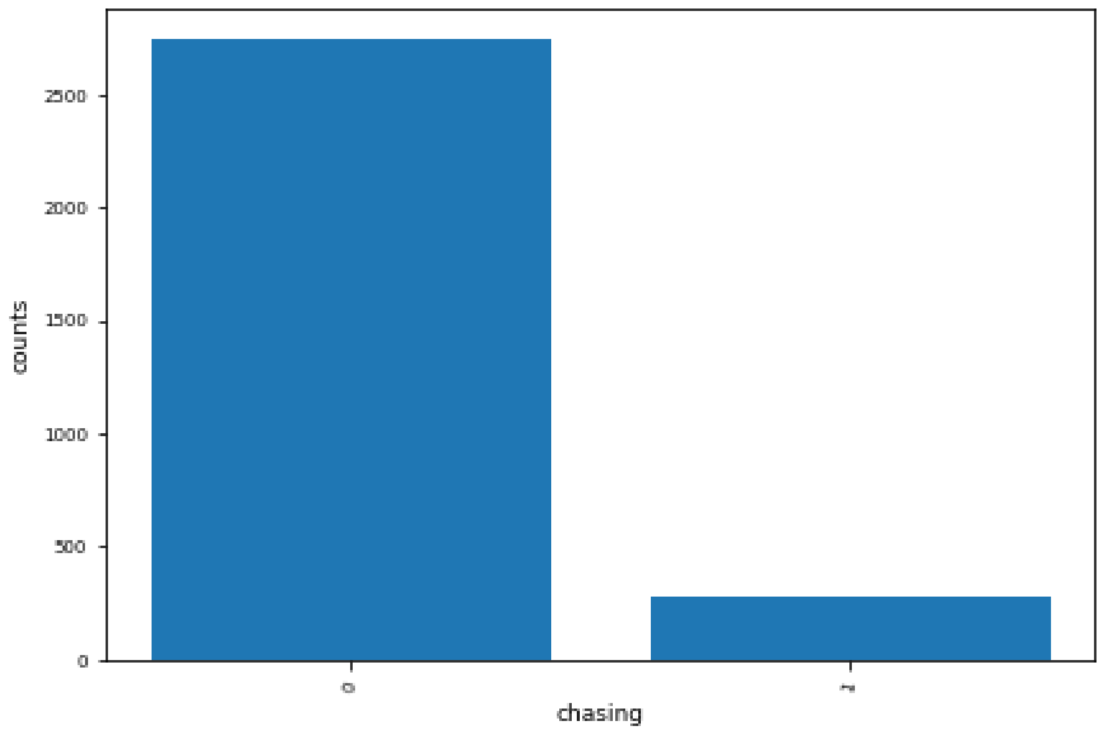
2500
2000
1500
1000
500
0
counts
Adult
Juvenile
Unknown
age

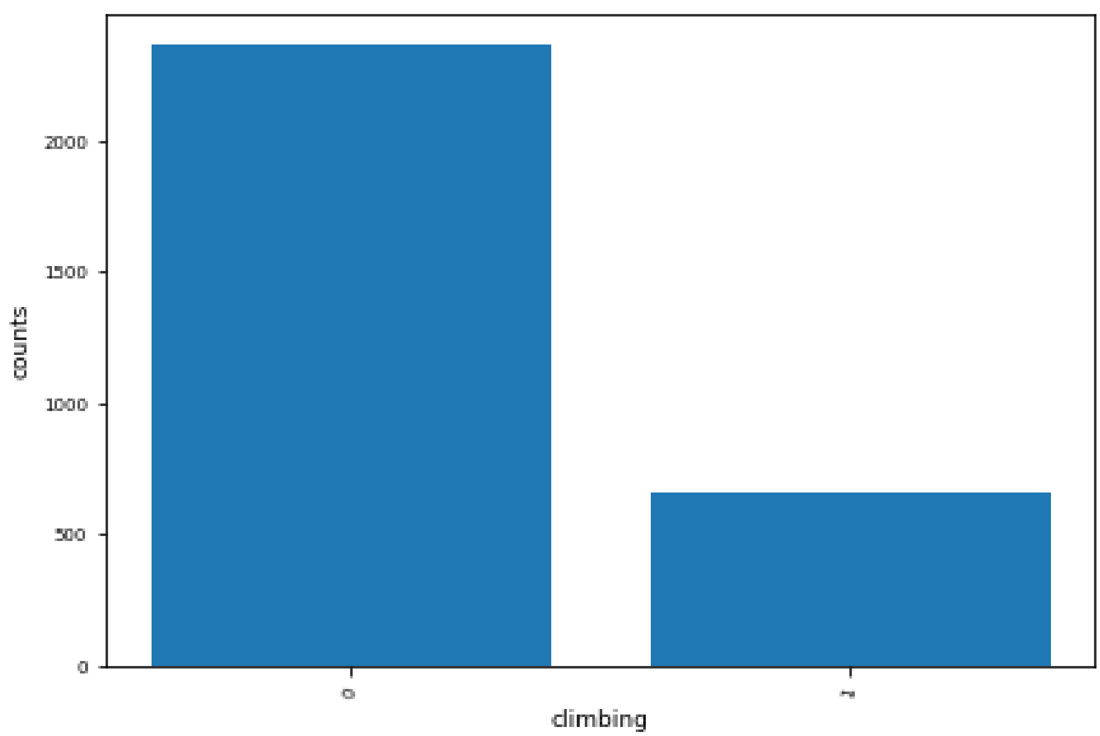


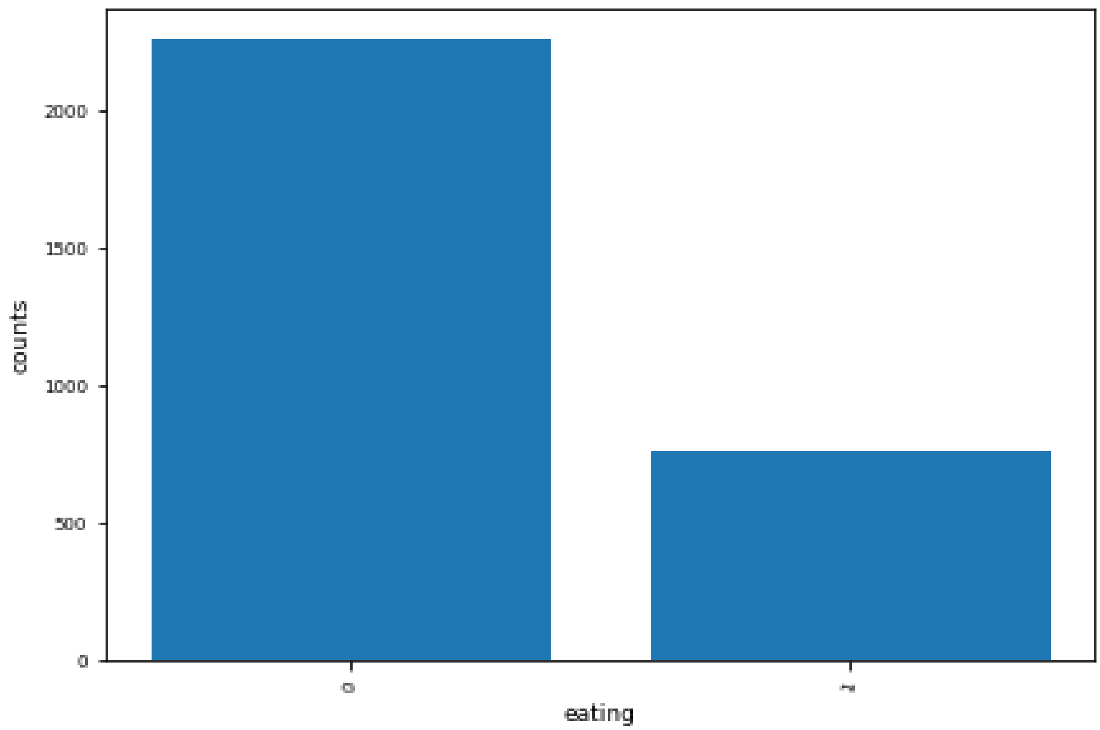
2500
2000
1500
1000
500
0
counts
Gray
Cinnamon
Black
primary_fur_color


800
700
600
500
400
300
200
100
0
counts
Cinnamon
White
Cinnamon, White
Gray
Gray, White
Black
Black, Cinnamon, White
Black, White
Black, Cinnamon
Gray, Black
highlight_fur_color

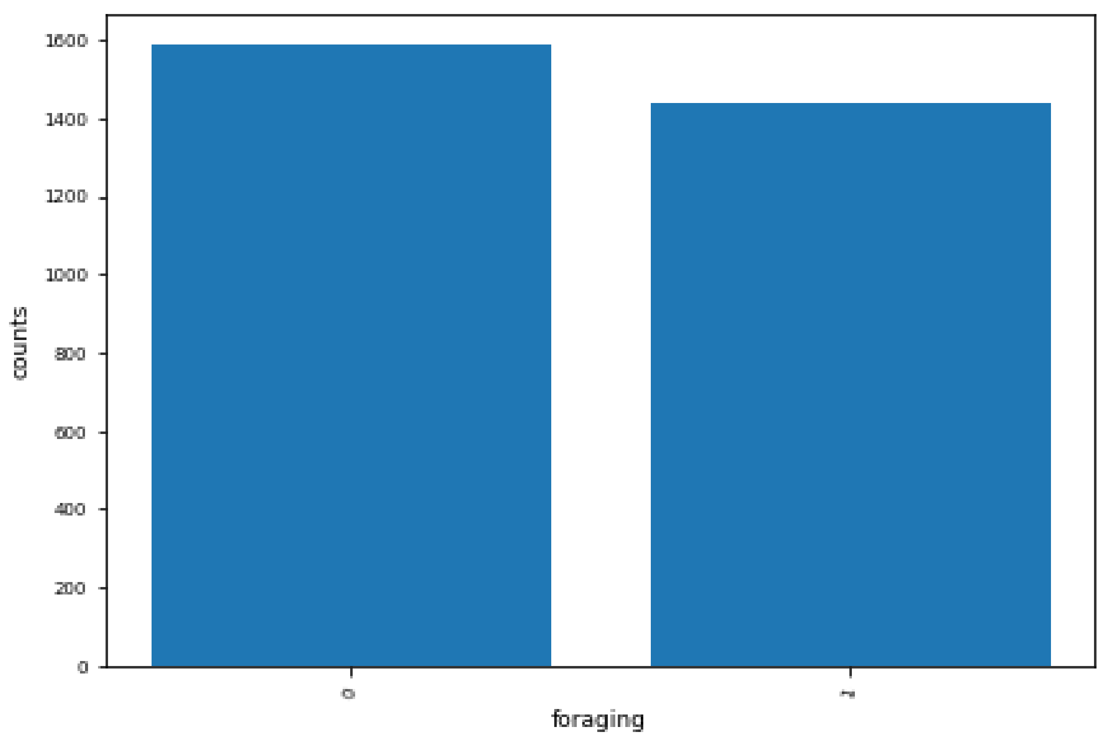

2000
1500
1000
500
0
counts
0
1
running


2500
2000
1500
1000
500
0
counts
0
1
chasing

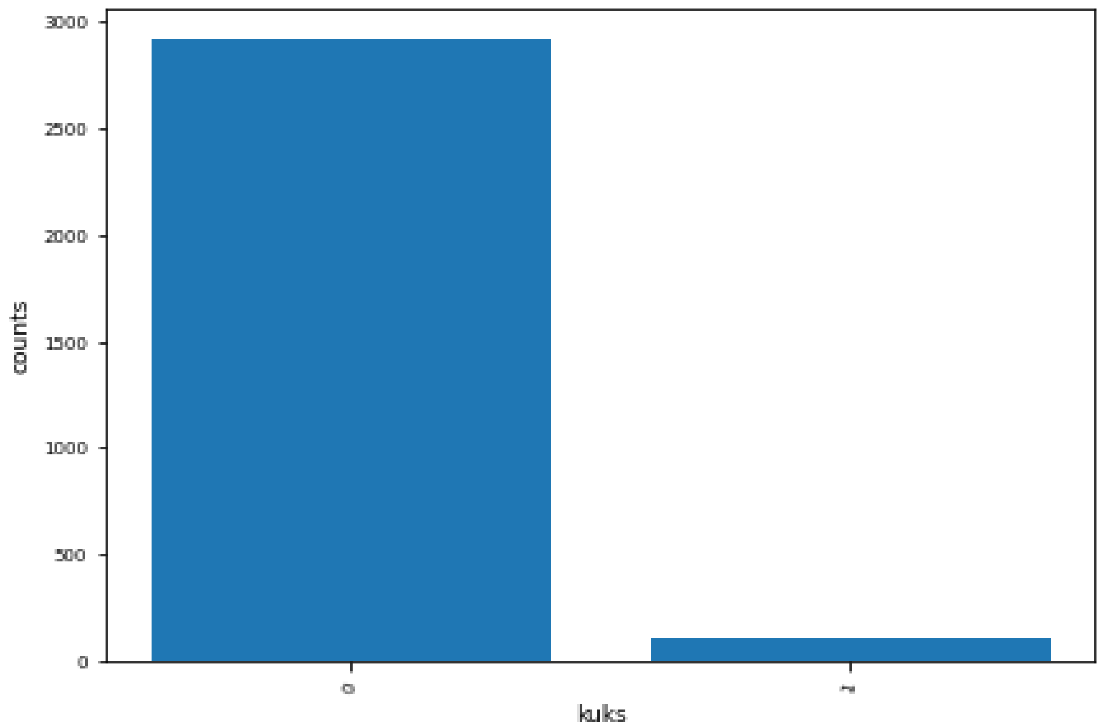

2000
1500
1000
500
0
counts
0
1
climbing

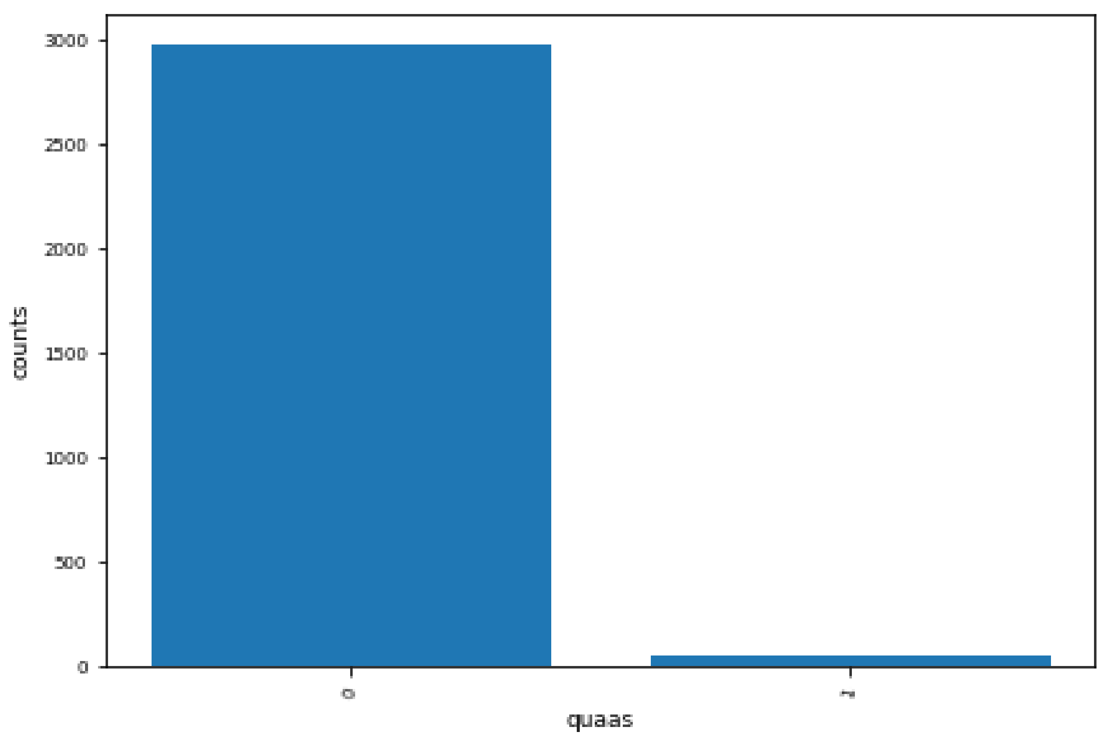


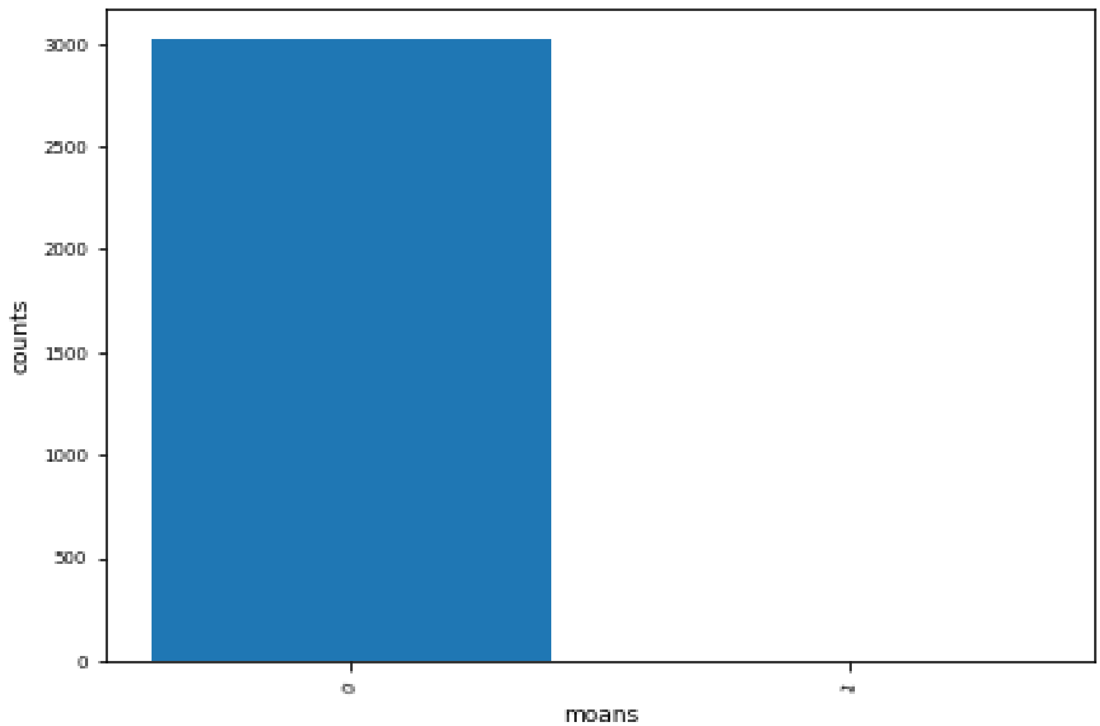
counts
2000
1500
1000
500
0
0
1
eating



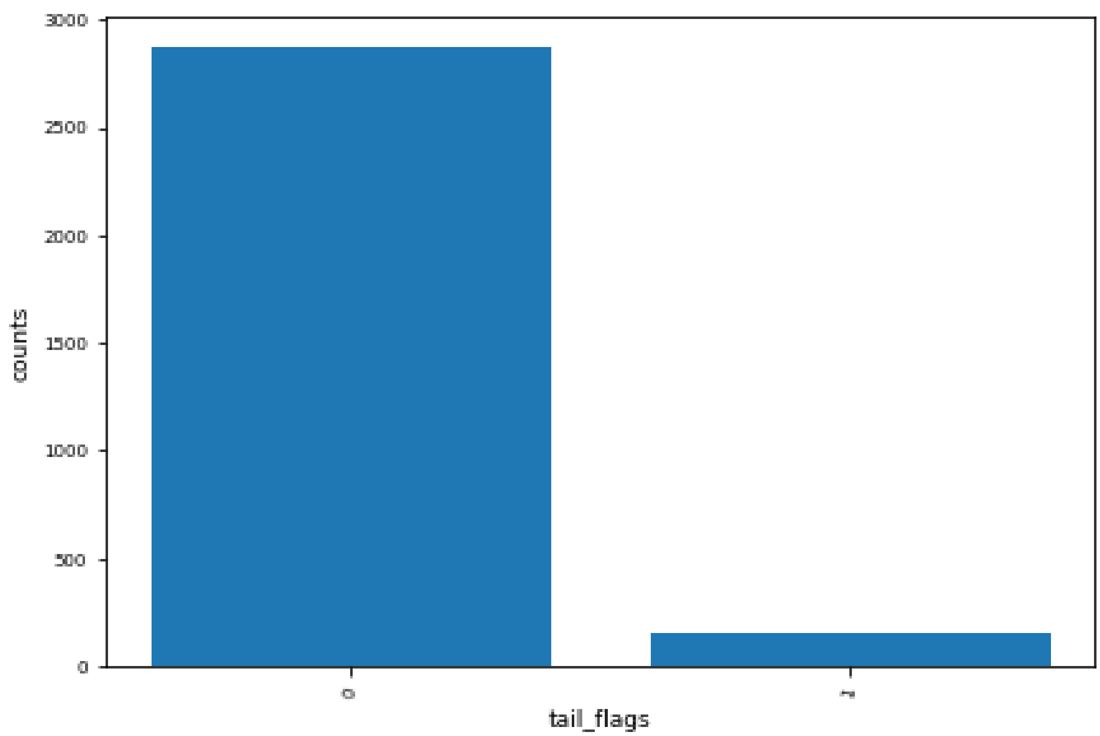
counts
1600
1400
1200
1000
800
600
400
200
0
0
1
foraging


3000
2500
2000
1500
1000
500
0
counts
0
1
kuks

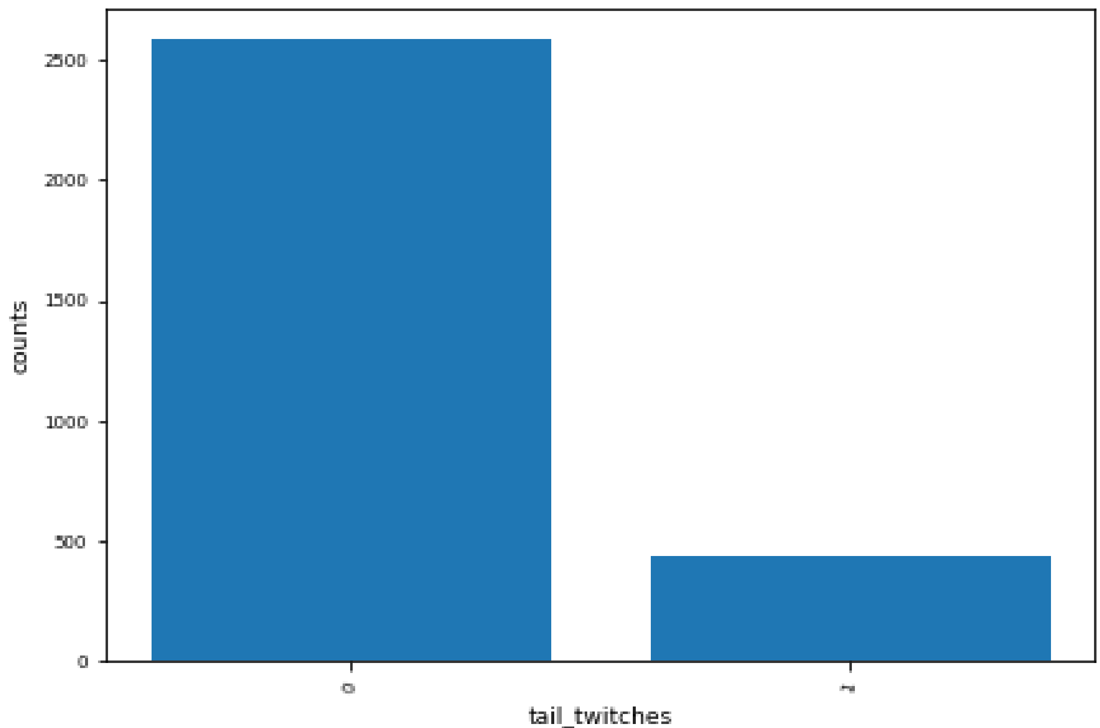

3000
2500
2000
1500
1000
500
0
counts
0
1
quaas



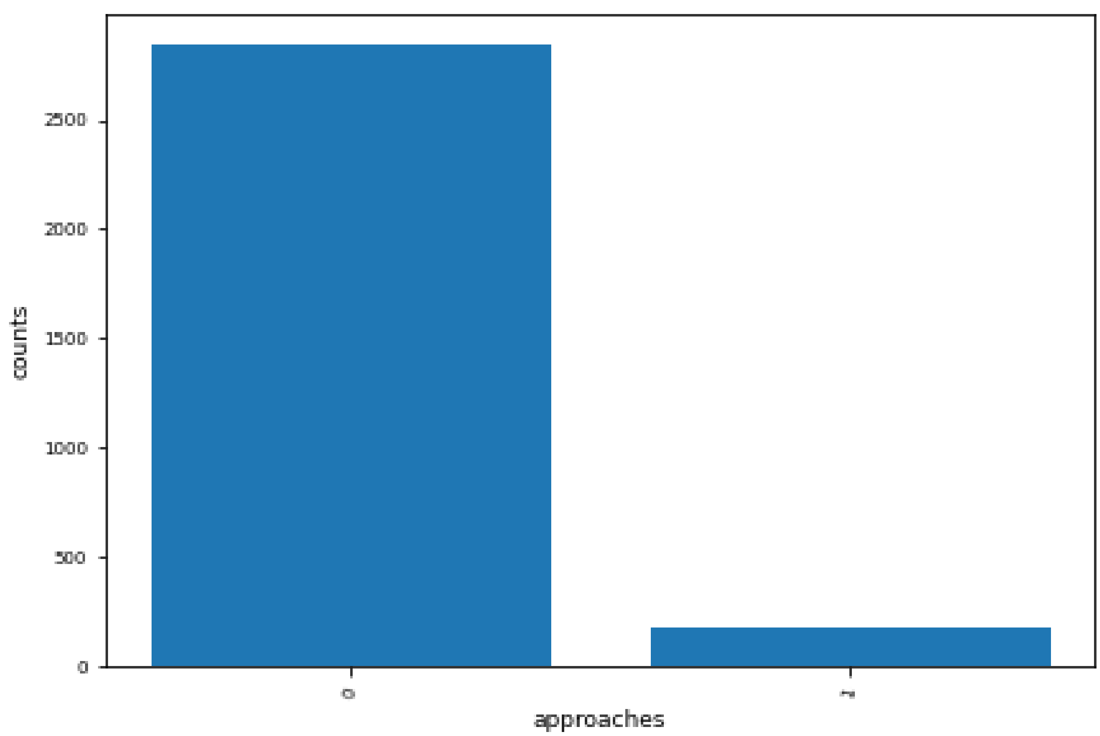
3000
2500
2000
1500
1000
500
0
counts
0
1
moans



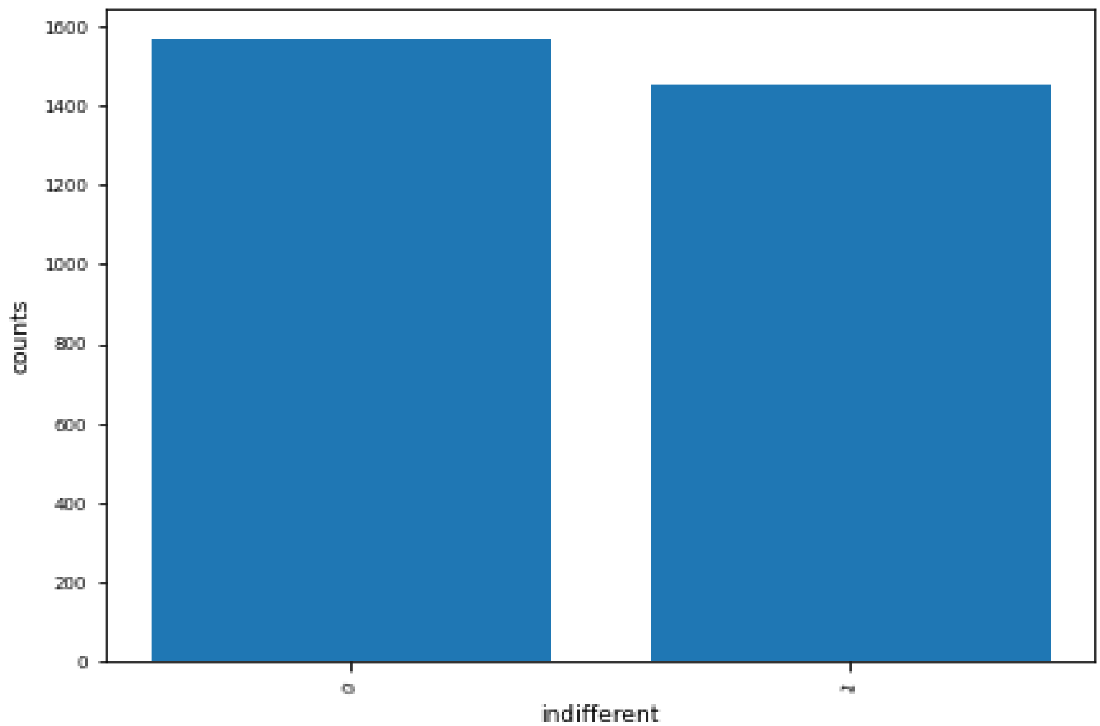
3000
2500
2000
1500
1000
500
0
counts
0
1
tail_flags

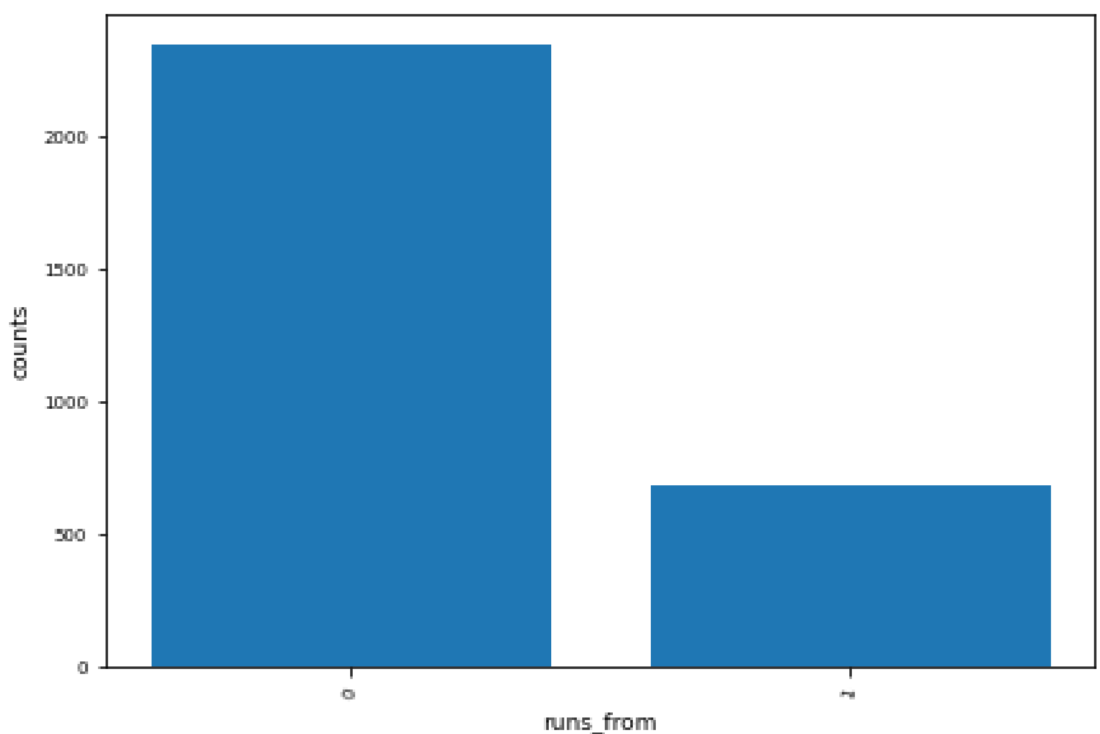


2500
2000
1500
1000
500
0
counts
0
1
tail_twitches



2500
2000
1500
1000
500
0
counts
0
1
approaches





Squirrel Distributions (creating maps)

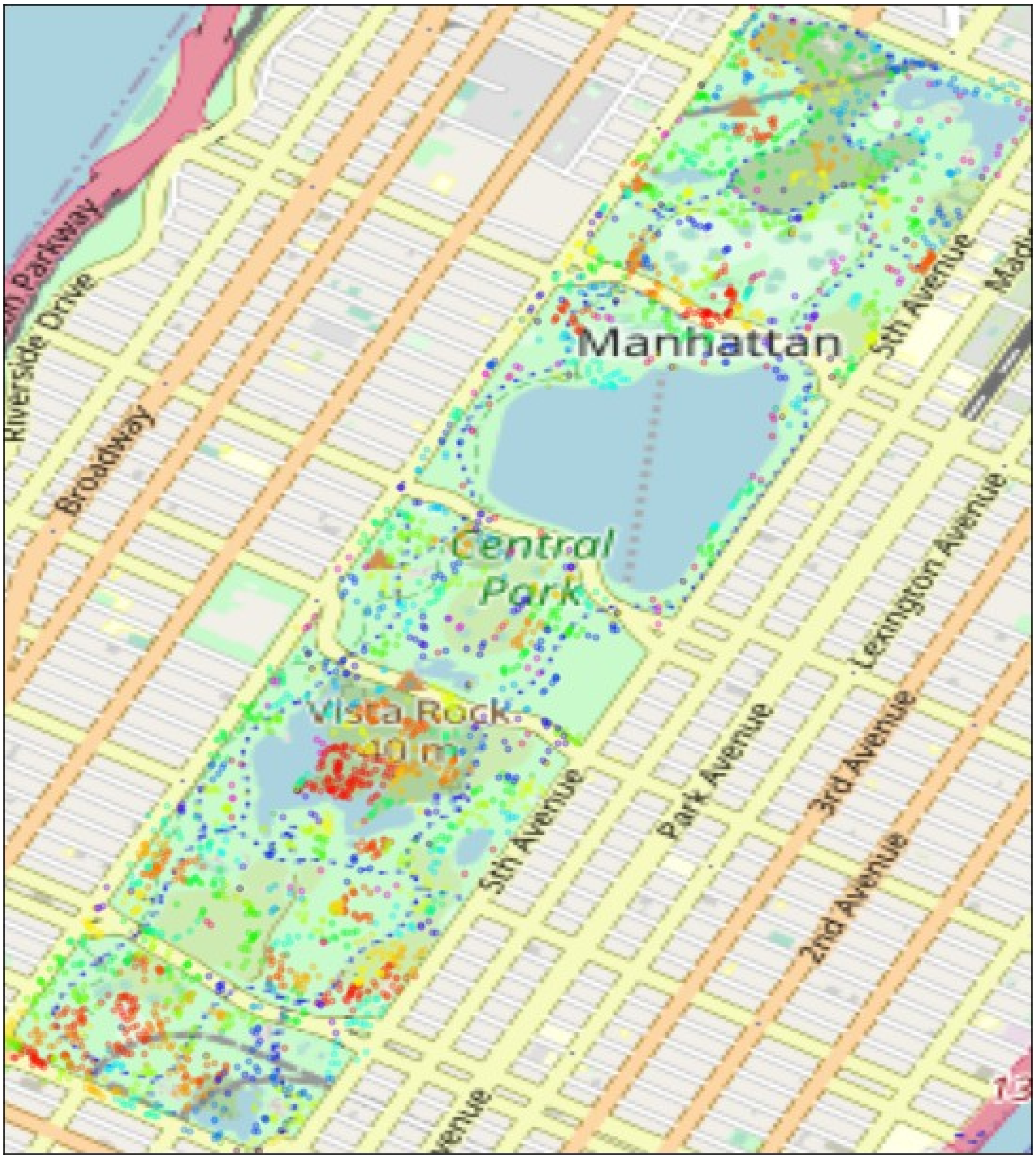
The first approach we chose was to visualize the squirrels on a map of Central Park. Our initial step was to simply add a dot for each squirrel at its latitude and longitude coordinates. A later stage with dots coloured according to squirrel color can be seen. It makes obvious the prevalence of the Eastern grey squirrel (Sciurus carolinensis). Other maps showed the Time of Day the squirrel was sighted, the date or whether it was a juvenile or an adult. Notable information that can be gleaned from this is that squirrels particular seem to favour the area around Vista Rock. This is possibly due to it being a spot where people frequently stop for some time, meaning more food is available, either from visitors or from the trash. Another important observation can be even though the most populated area is the Vista Rock, it is not as much as populated by black squirrels.

```
plot_maps(nyc_squirrels, save_fig=False)
interest_list = ['hectare', 'slot', 'date', 'age',
'primary_fur_color',
       'highlight_fur_color', 'running', 'chasing', 'climbing',
'eating',
       'foraging', 'kuks', 'quaas', 'moans', 'tail_flags',
       'tail_twitches', 'approaches', 'indifferent', 'runs_from']
for c in interest_list:
  plot_maps(nyc_squirrels, plotted_feature=c, save_fig=False)
```

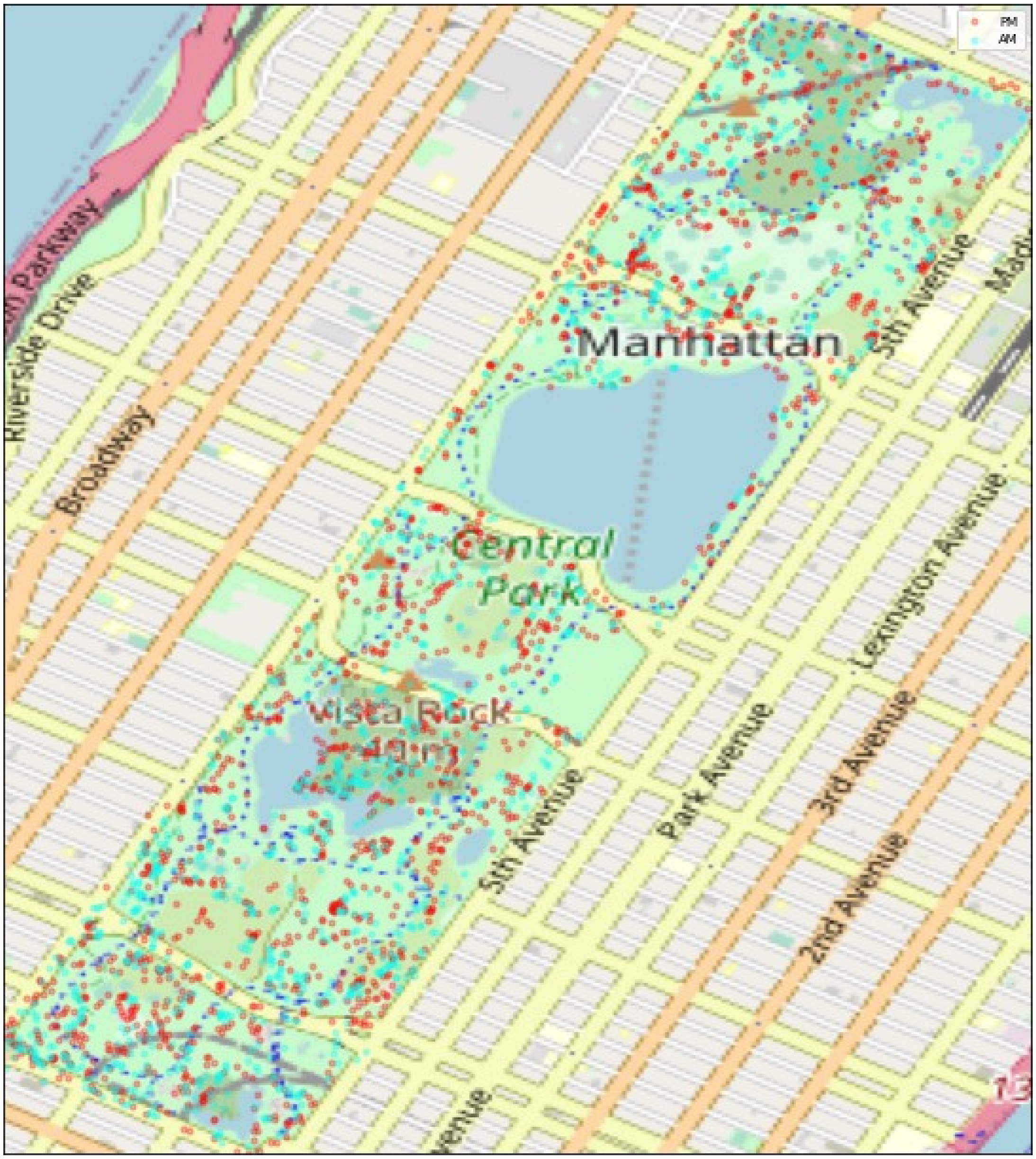
all


Riverside Drive
Broadway
Manhattan
Central Park
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

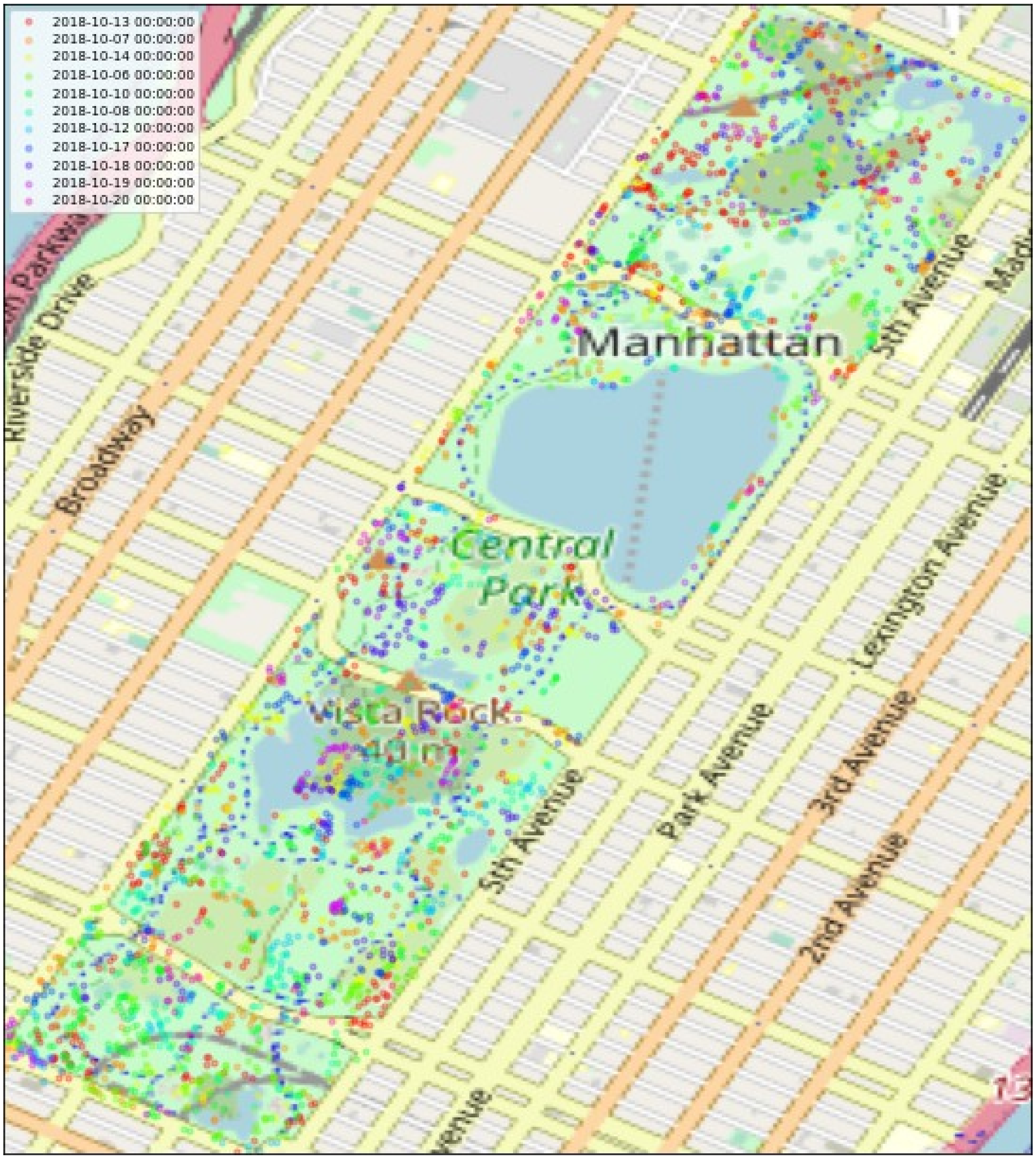
hectare

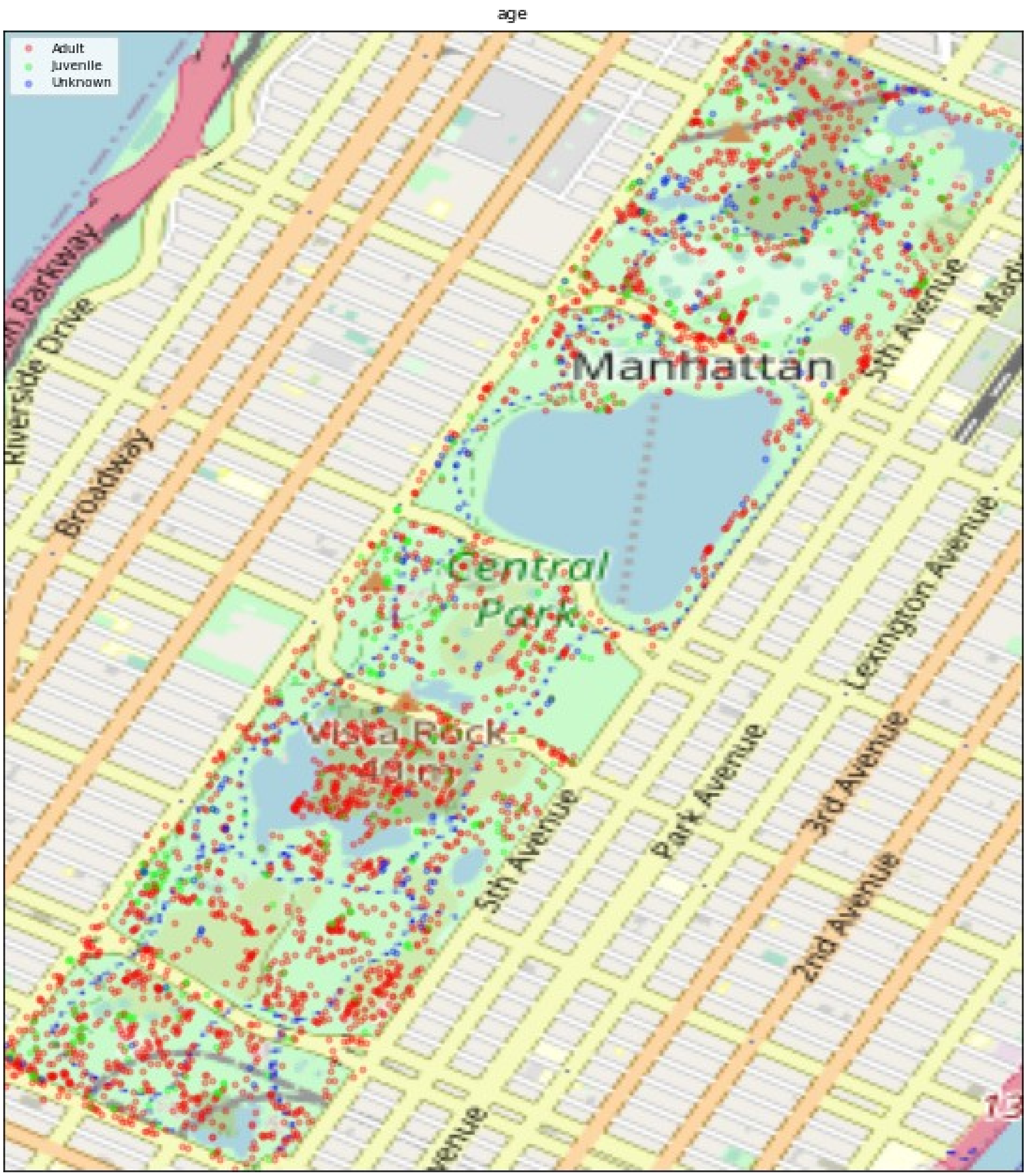

Riverside Drive
Broadway
Manhattan
Central
Park
Vista Rock
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

slot

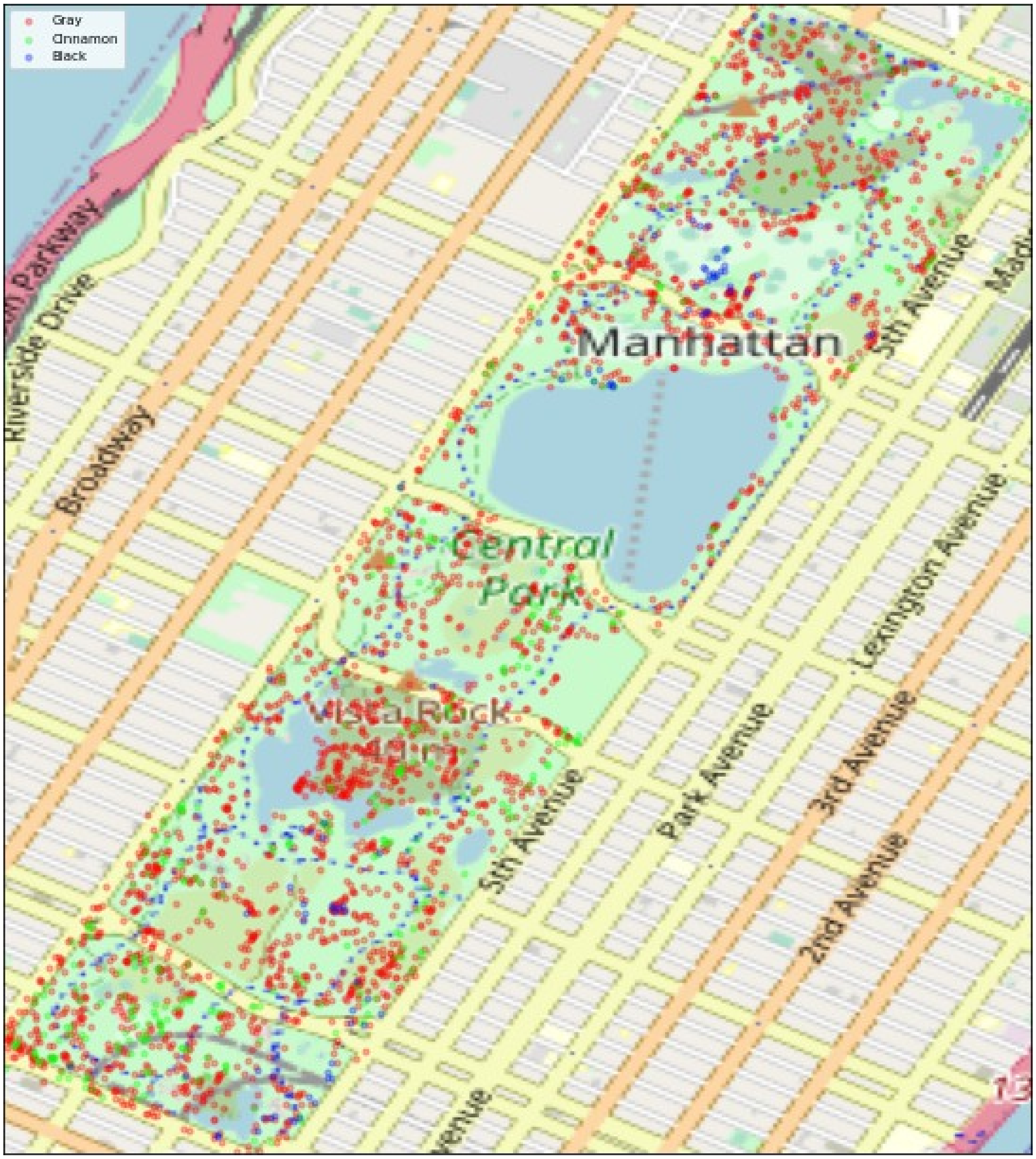

PM
AM
Manhattan
Central
Park
Vista Rock
Riverside Drive
Broadway
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

date

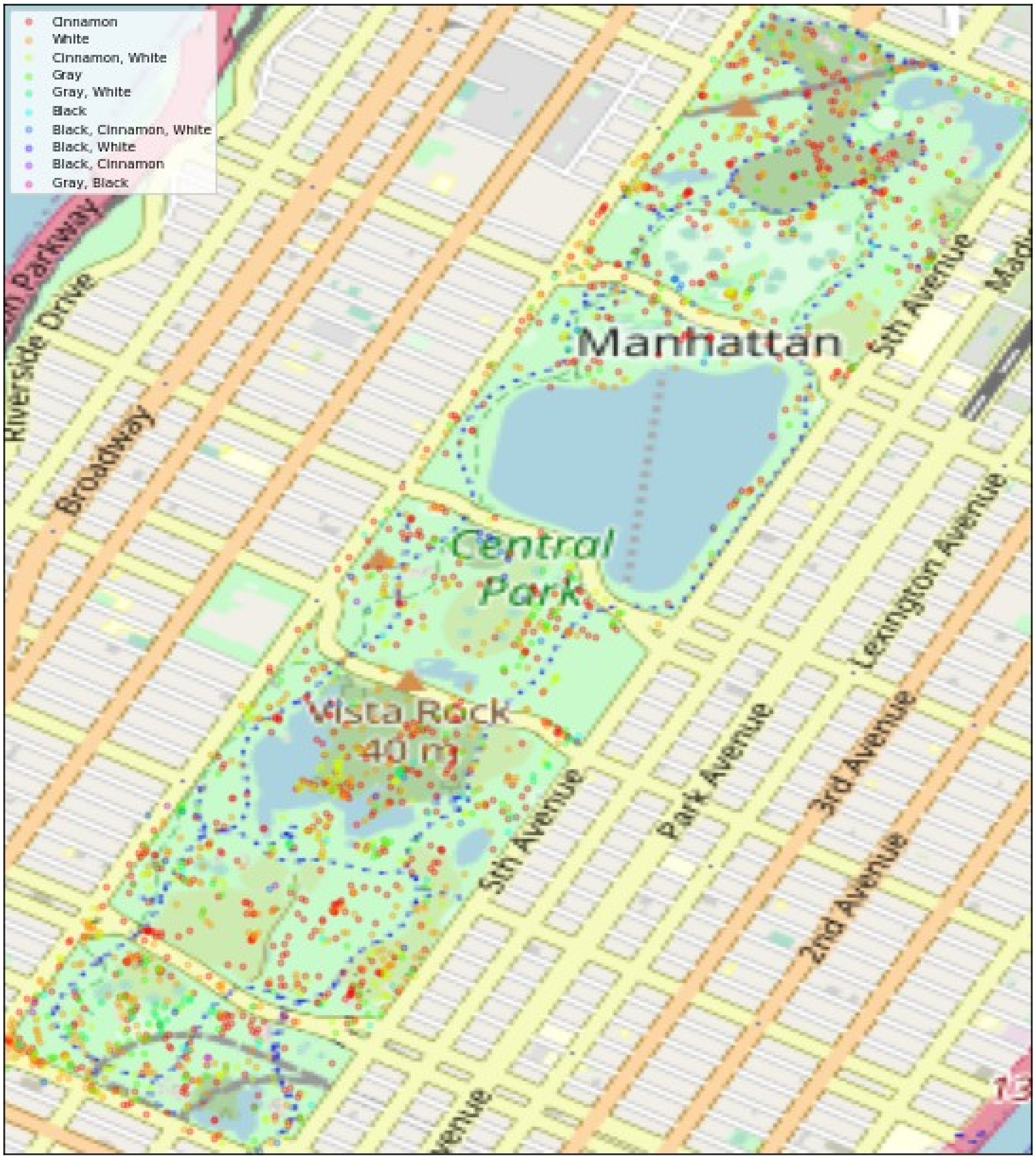

2018-10-13 00:00:00
2018-10-07 00:00:00
2018-10-14 00:00:00
2018-10-06 00:00:00
2018-10-10 00:00:00
2018-10-08 00:00:00
2018-10-12 00:00:00
2018-10-17 00:00:00
2018-10-18 00:00:00
2018-10-19 00:00:00
2018-10-20 00:00:00
Riverside Drive
Broadway
Manhattan
Central Park
Vista Rock
40 m
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue


age
Adult
Juvenile
Unknown
Riverside Drive
Broadway
Manhattan
Central Park
Vista Rock
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

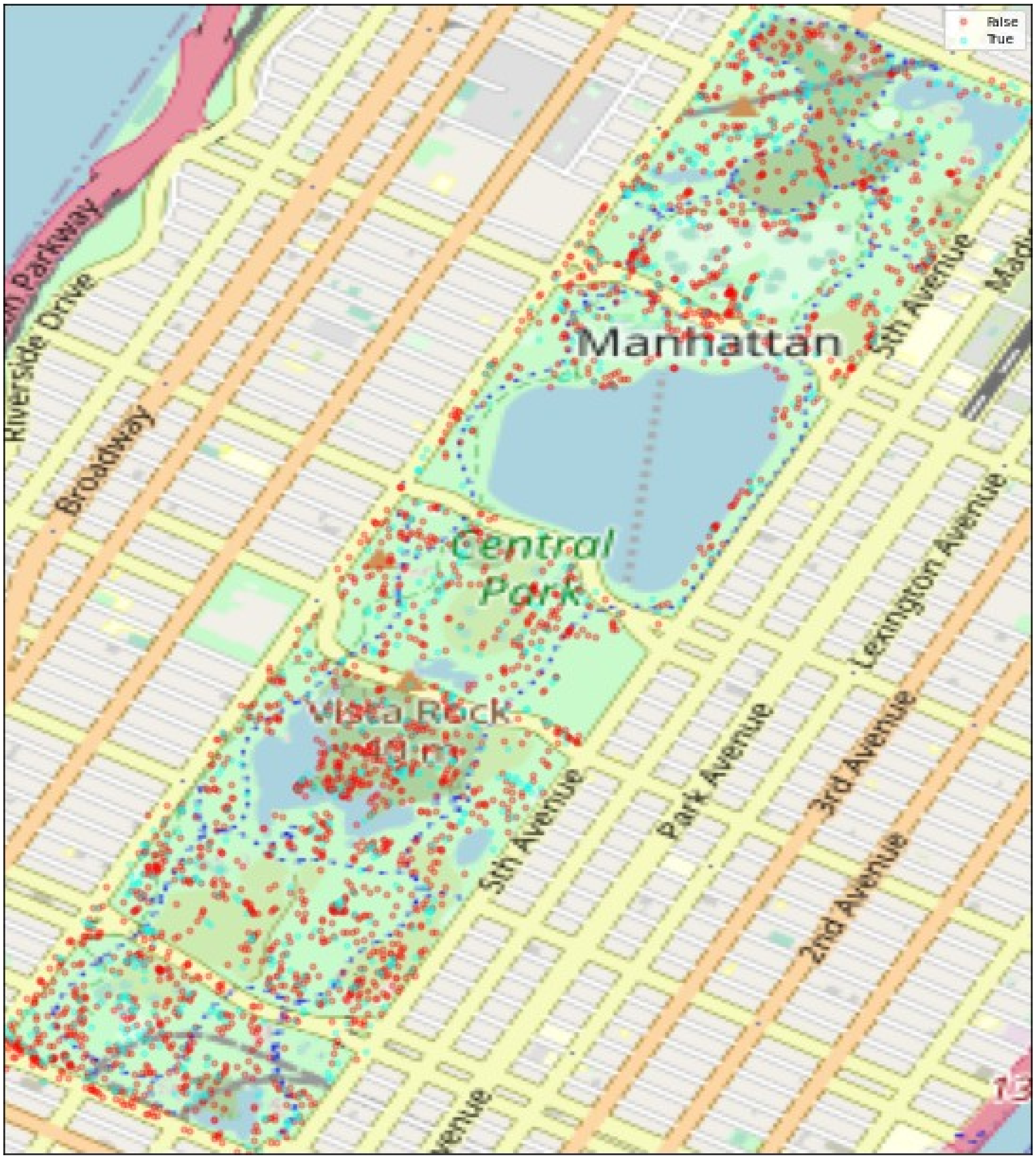
primary_fur_color



highlight_fur_color

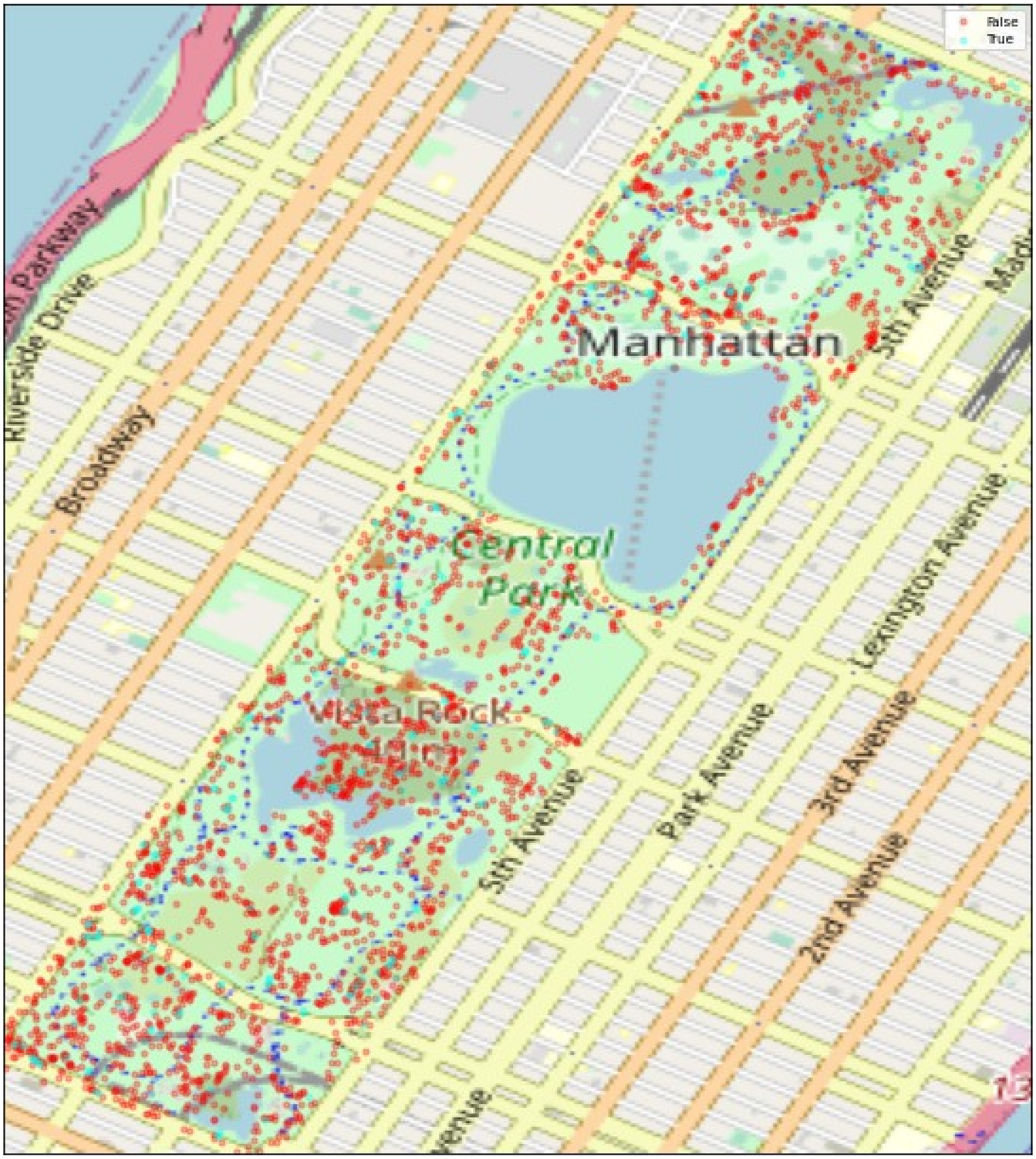


running

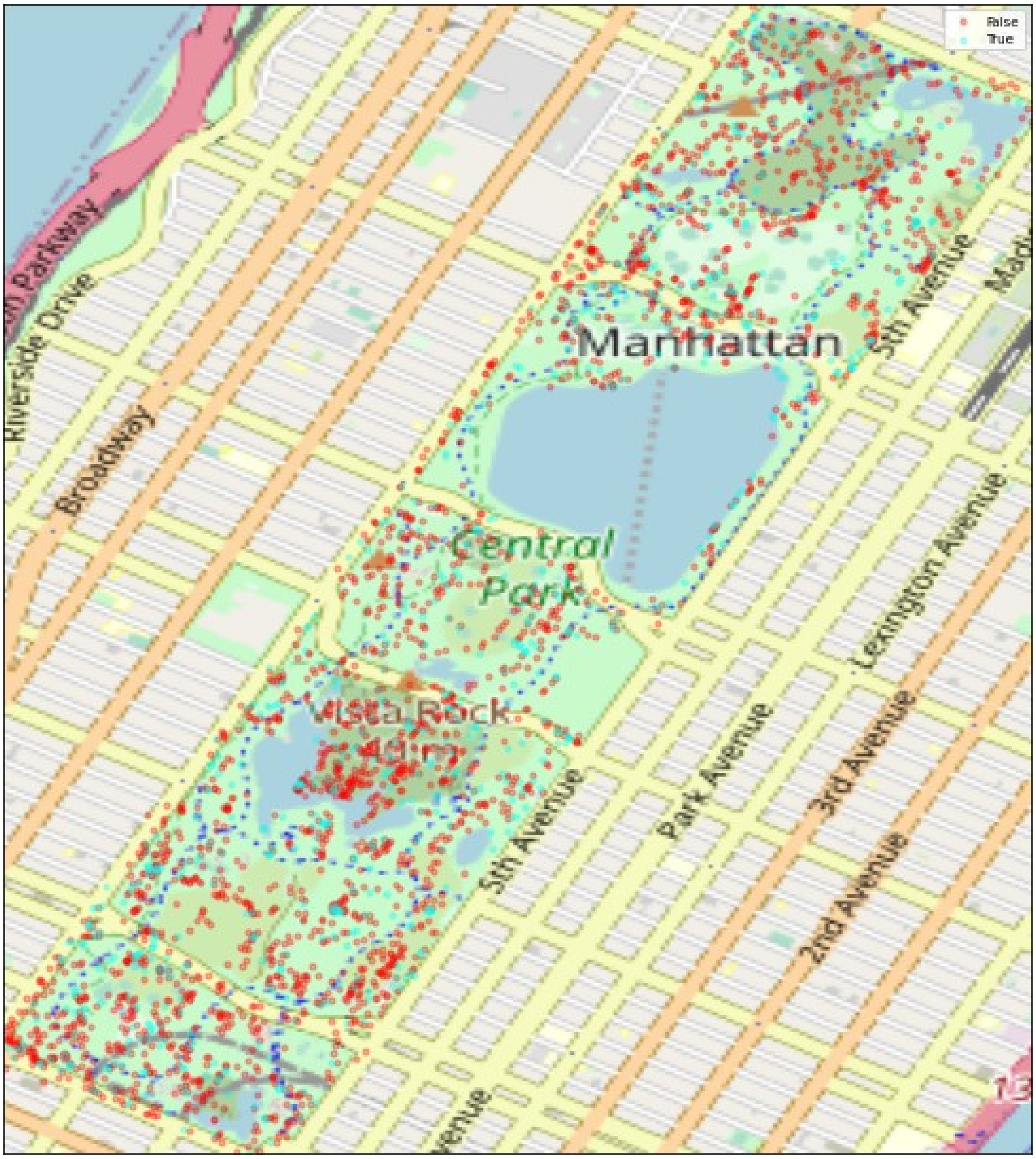


False
True
Manhattan
Central Park
Vista Rock
Riverside Drive
Broadway
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

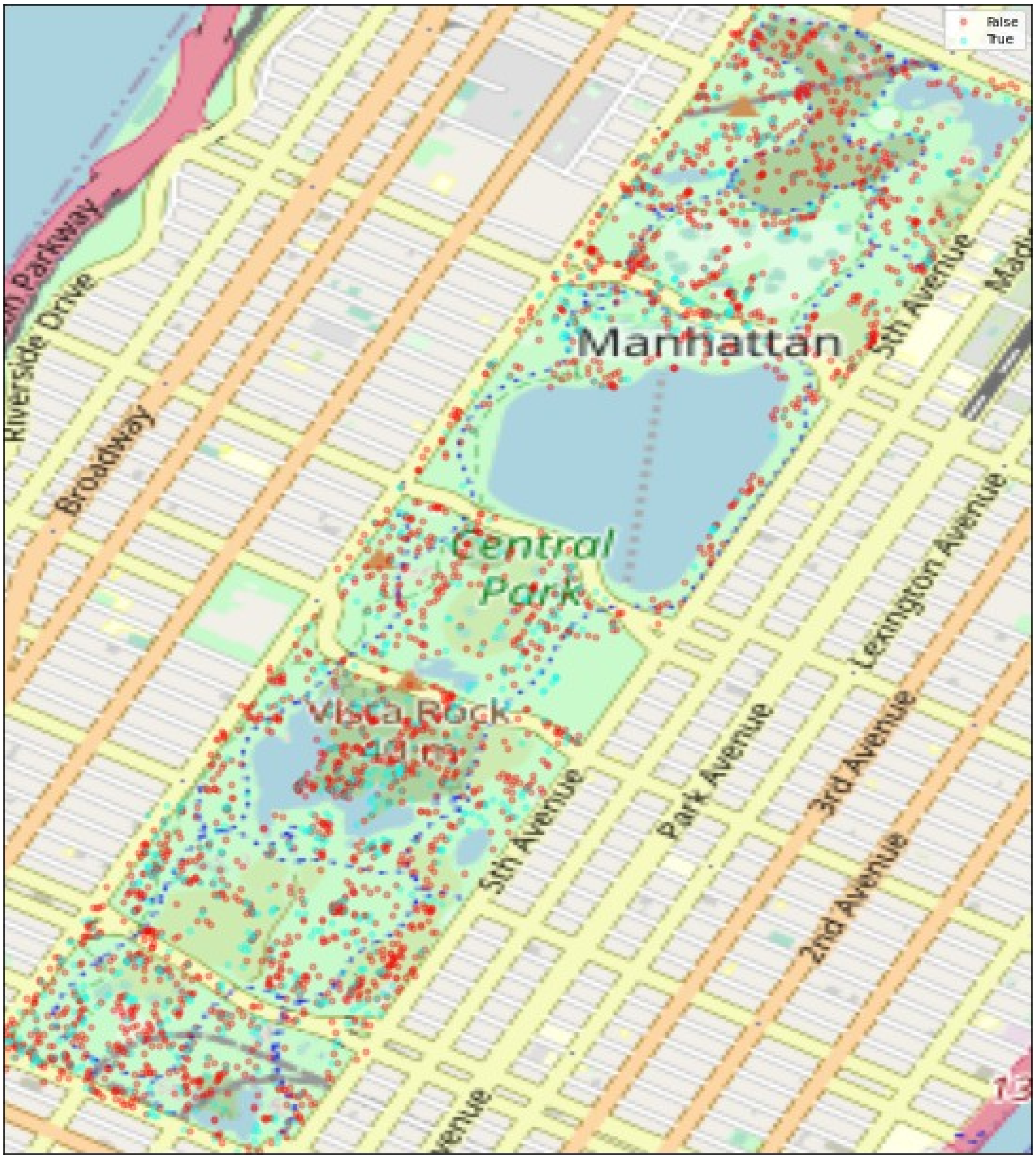
chasing


False
True
Riverside Drive
Broadway
Manhattan
Central Park
5th Avenue
Lexington Avenue
Park Avenue
3rd Avenue
2nd Avenue

climbing

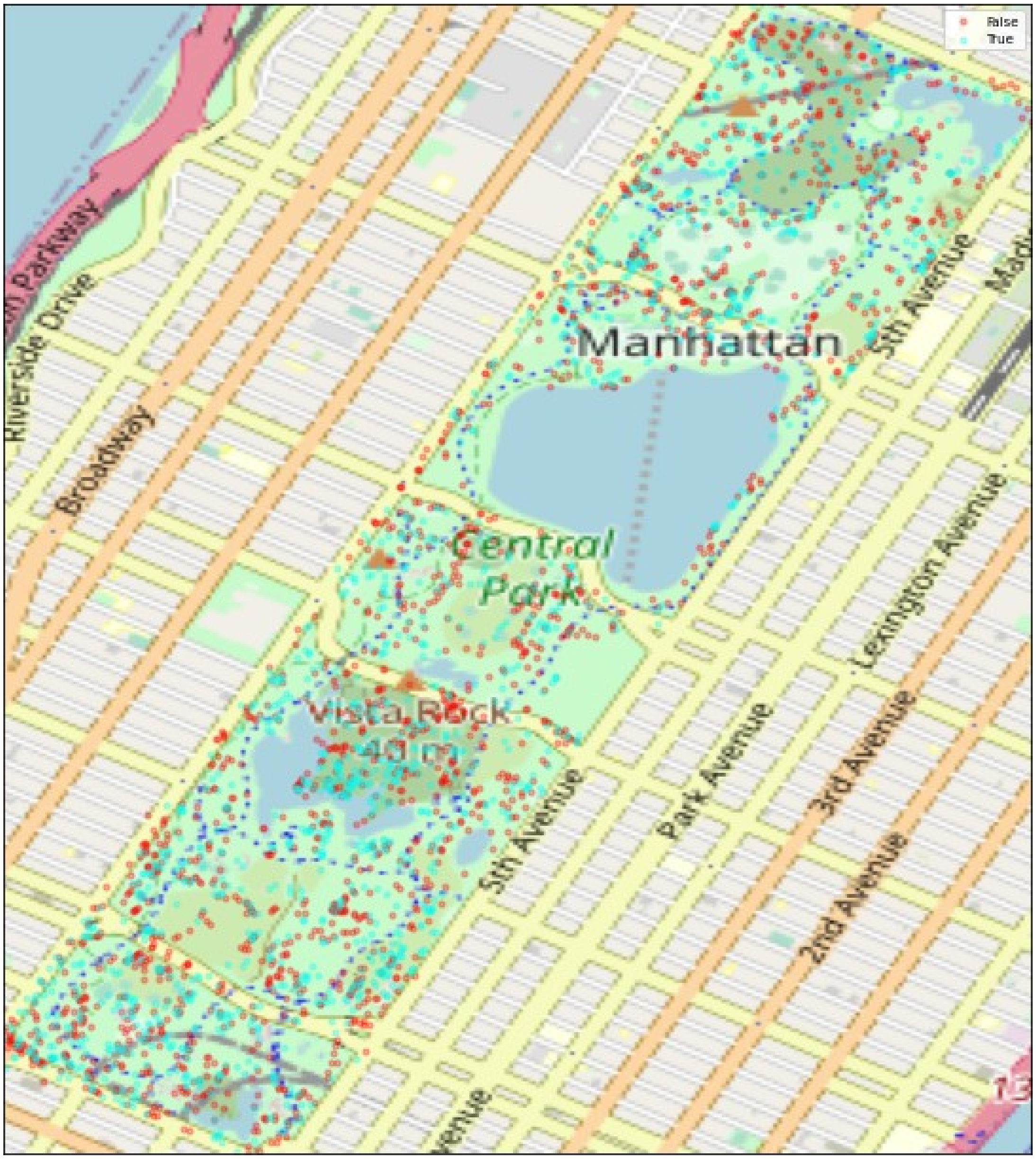

False
True
Manhattan
Central Park
Vista Rock
Riverside Drive
Broadway
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

eating

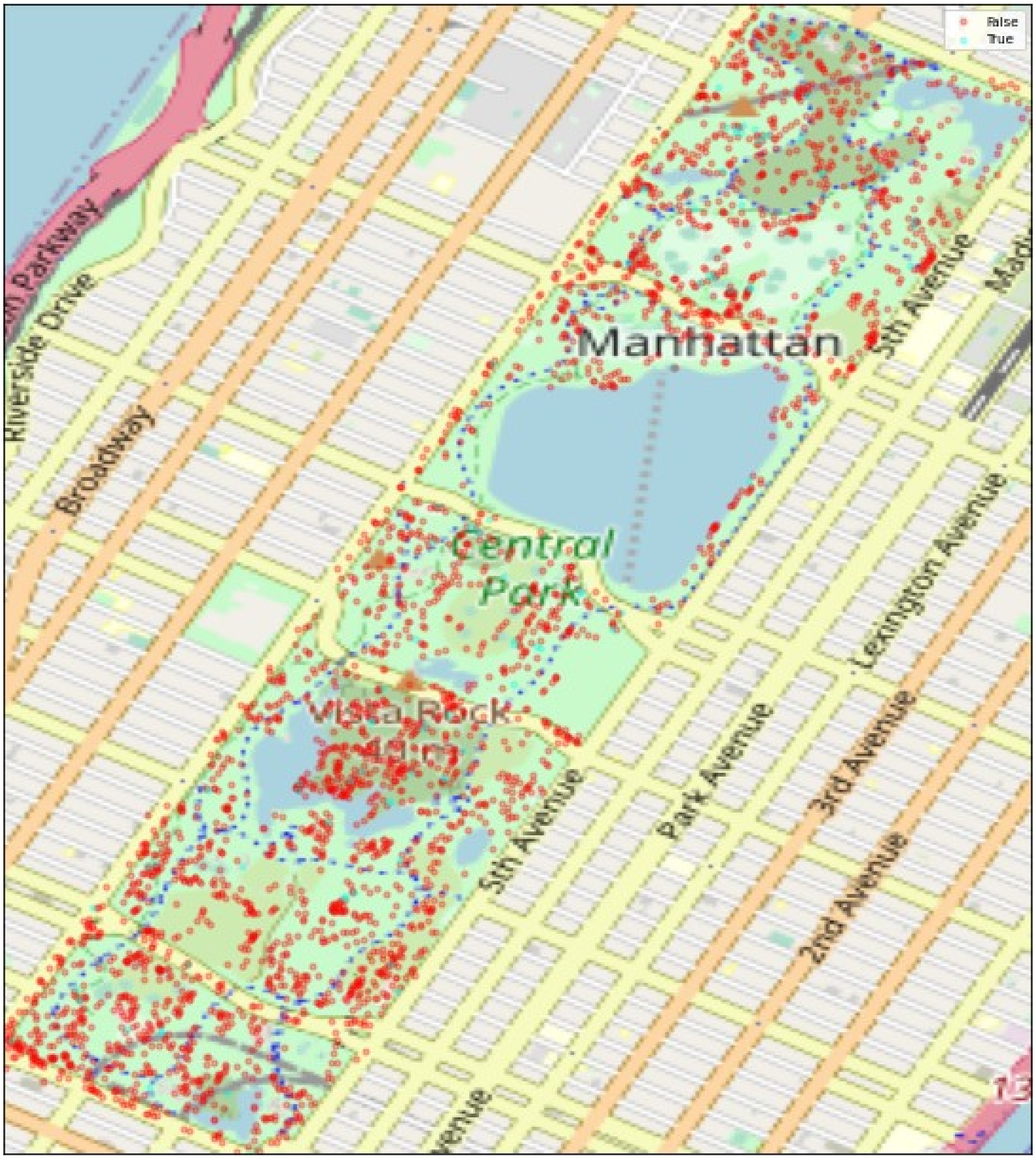


False
True
Riverside Drive
Broadway
Manhattan
Central Park
Vista Rock
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

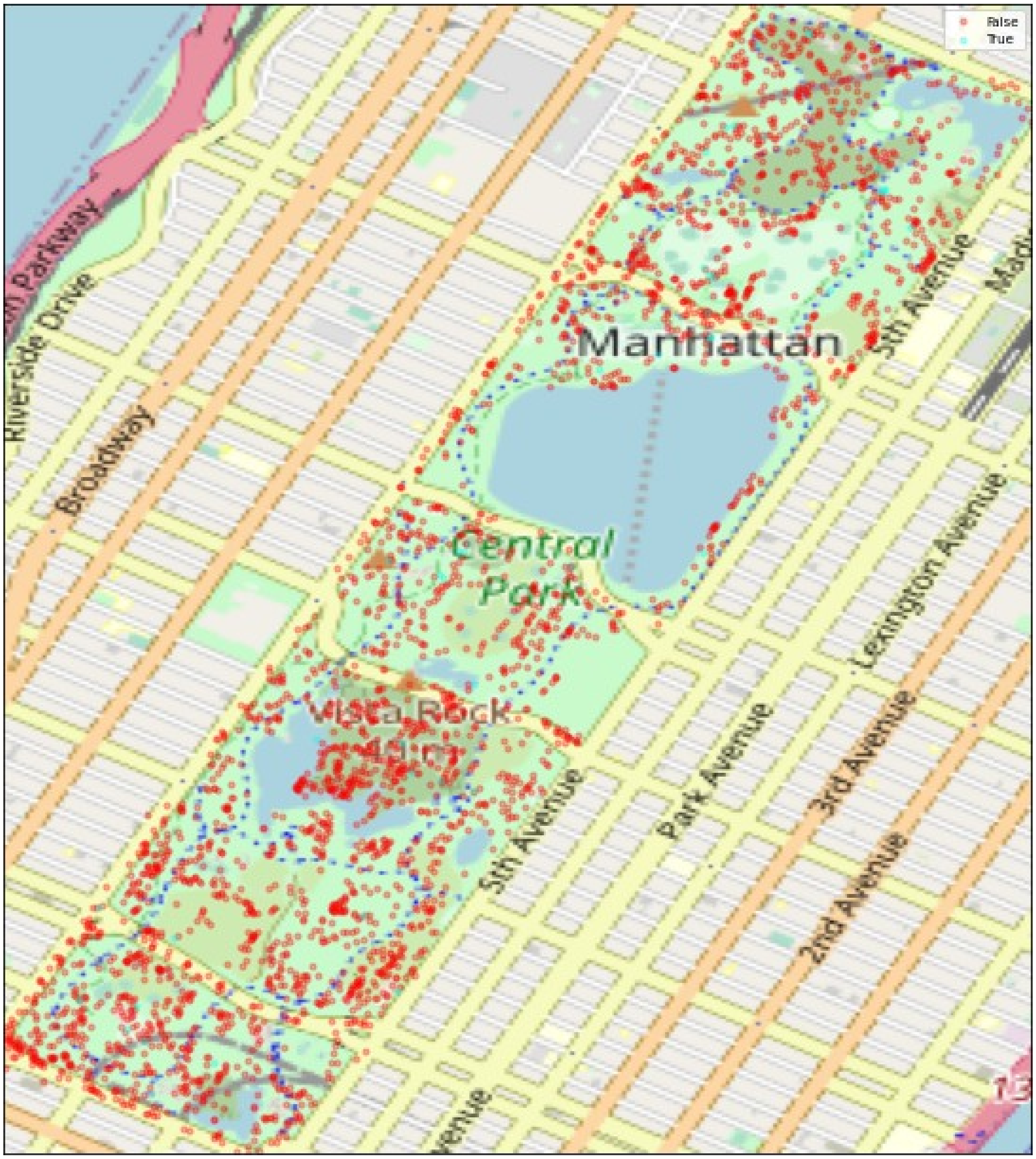
foraging


False
True
Riverside Drive
Broadway
Manhattan
Central Park
Vista Rock
40 m
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

kuks

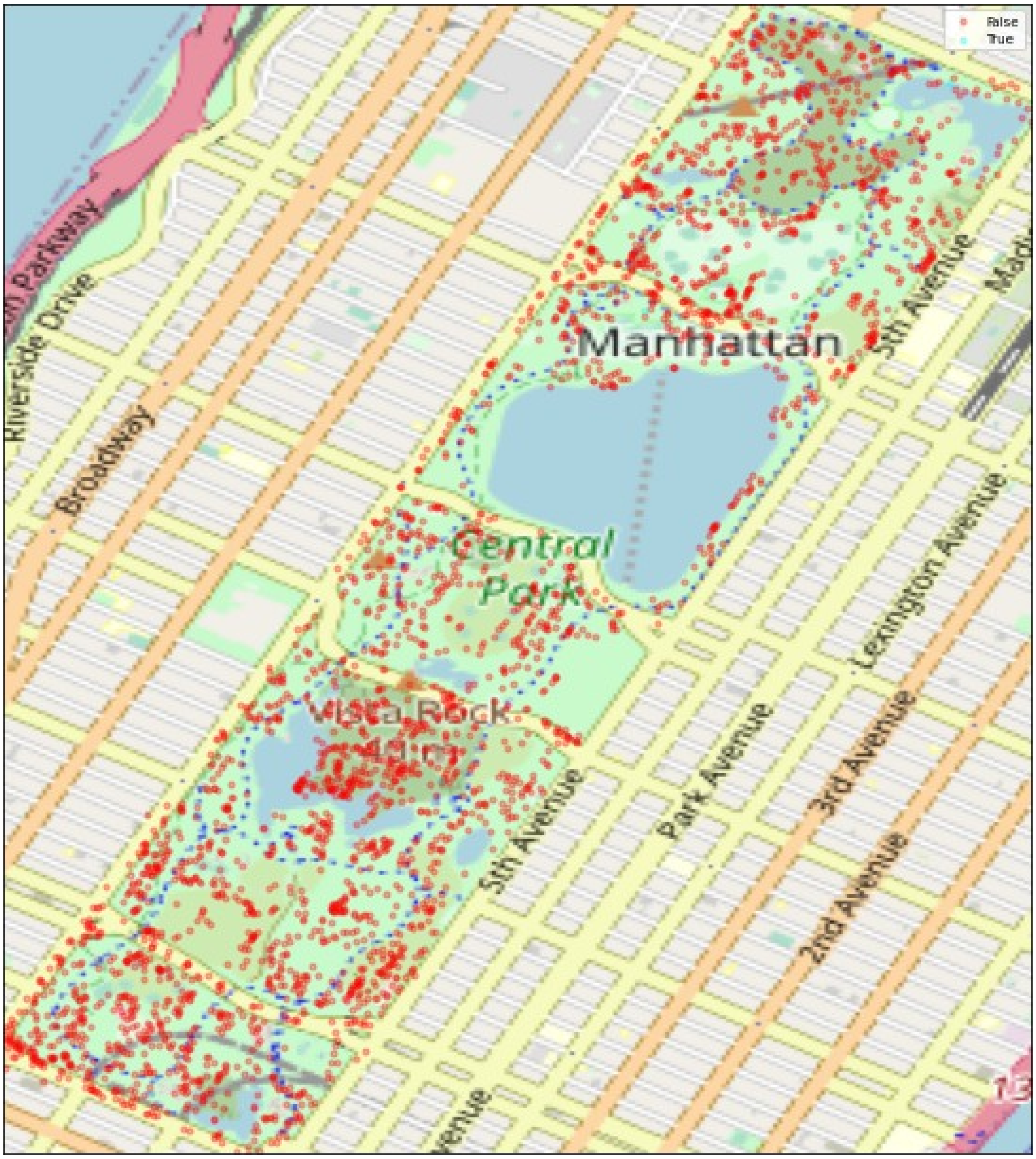

False
True
Manhattan
Central Park
Vista Rock
Riverside Drive
Broadway
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

quaas



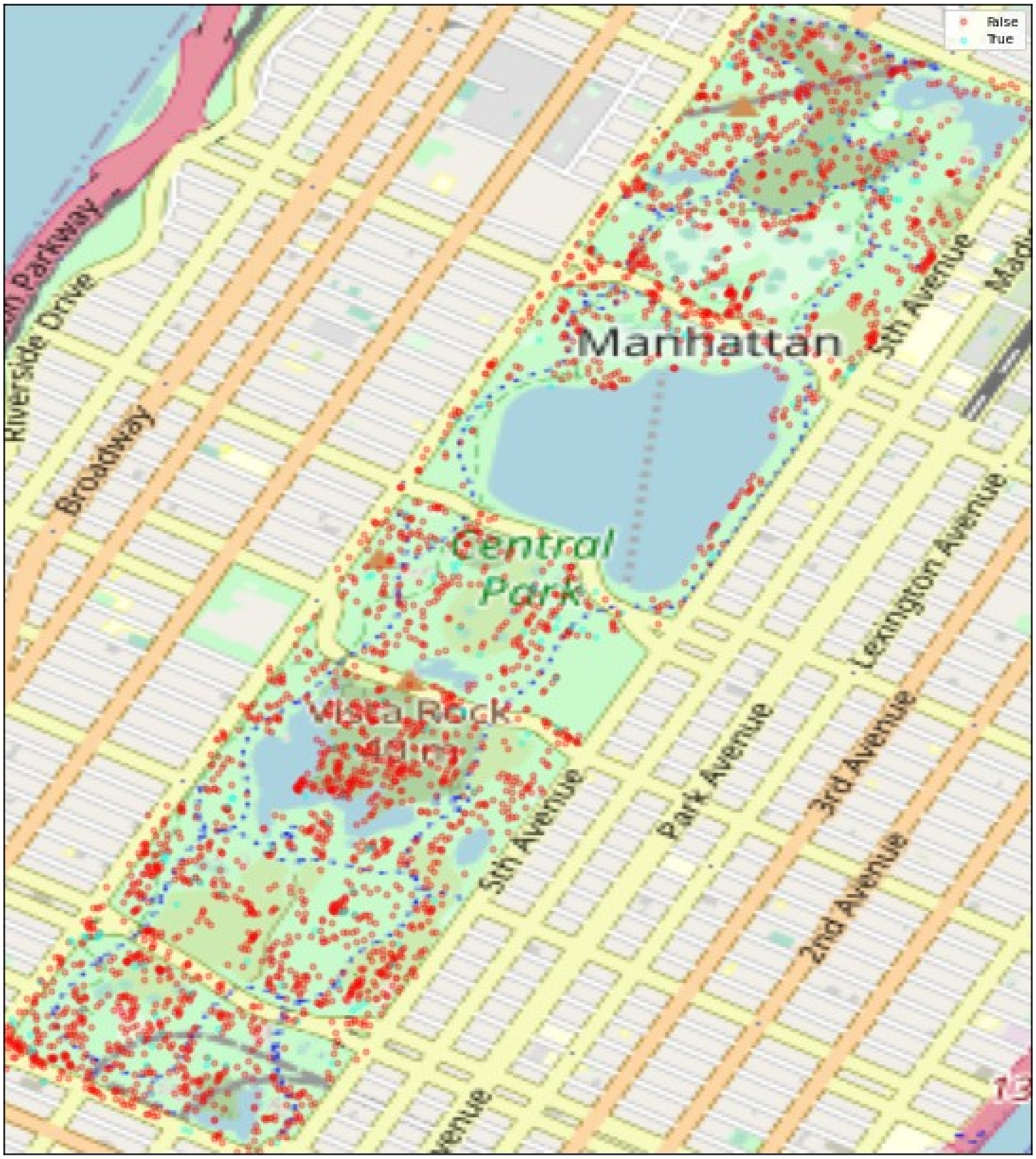
False
True
Riverside Drive
Broadway
Manhattan
Central Park
Vista Rock
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

moans



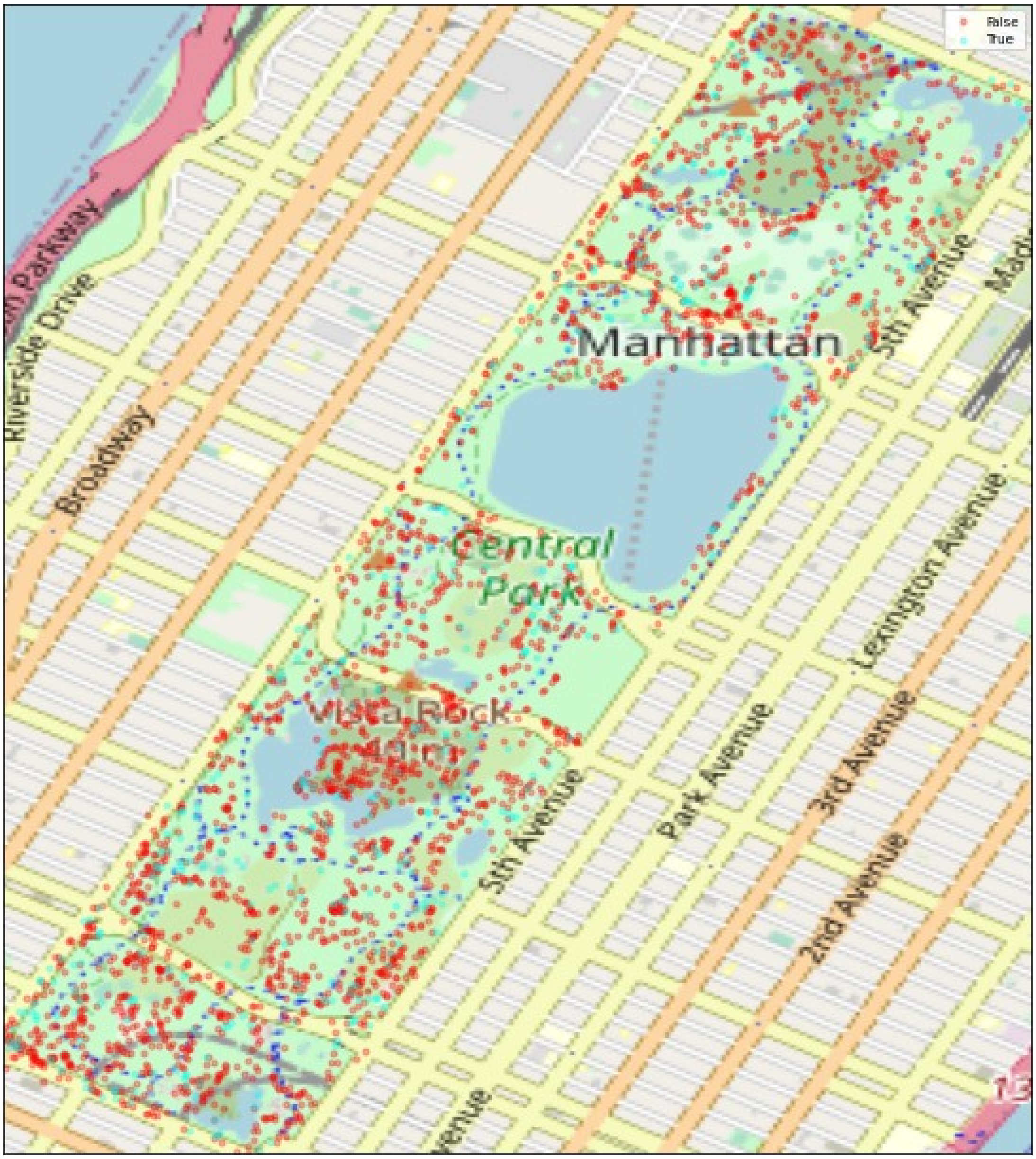
False
True
Riverside Drive
Broadway
Manhattan
Central Park
Vista Rock
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

tail_flags

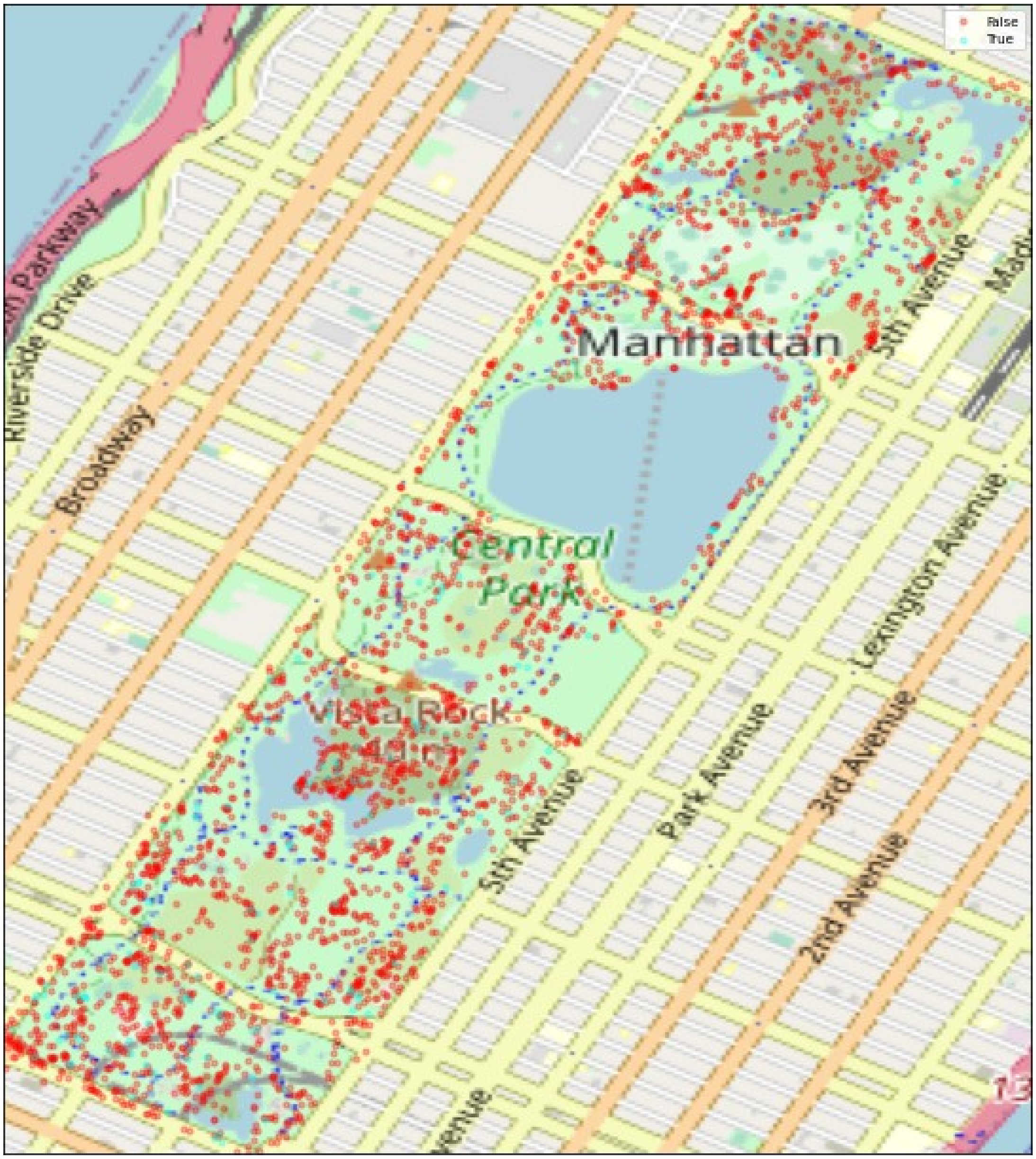


False
True
Manhattan
Central Park
Vista Rock
Riverside Drive
Broadway
5th Avenue
Park Avenue
Lexington Avenue
3rd Avenue
2nd Avenue

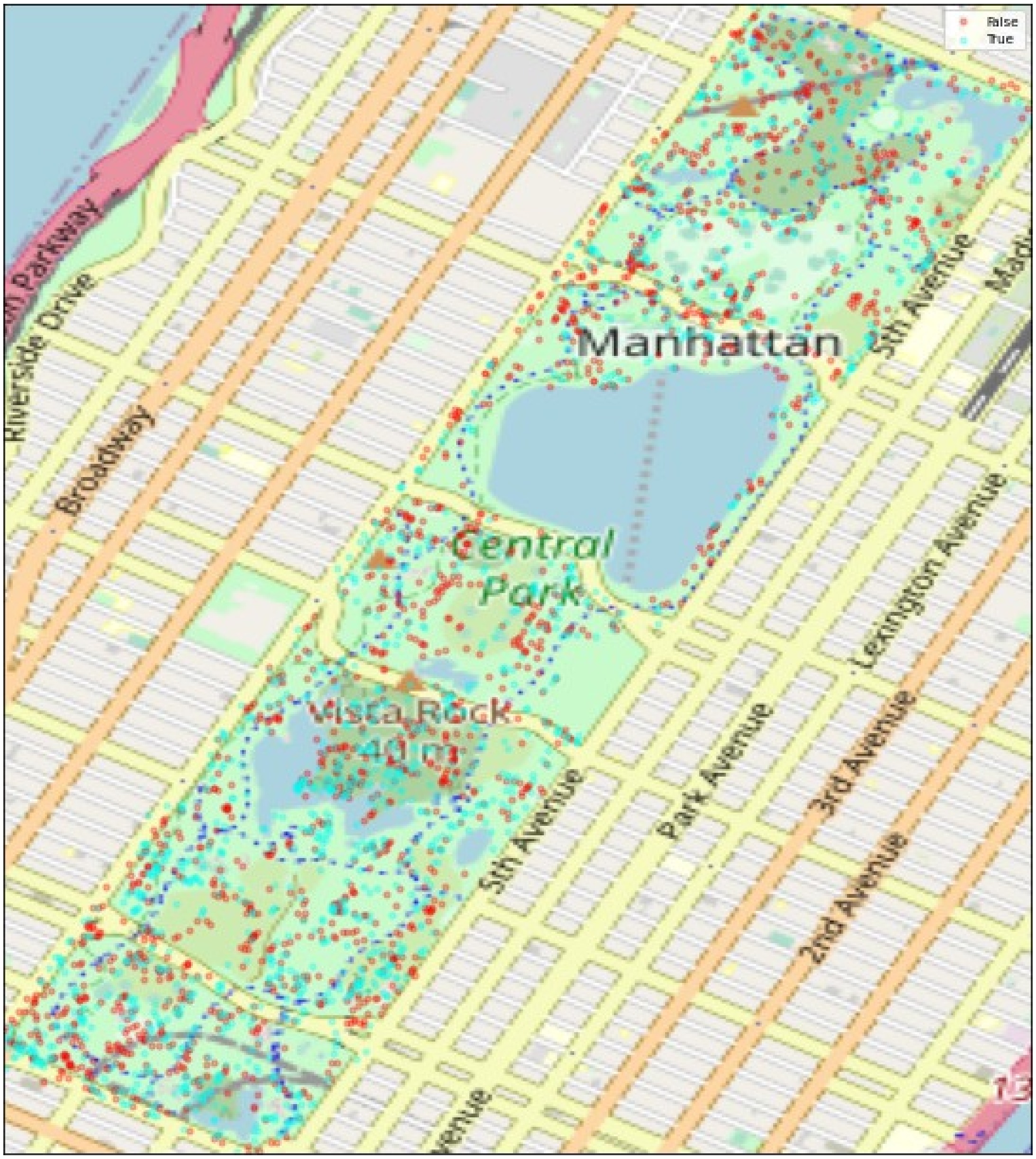
tail_twitches

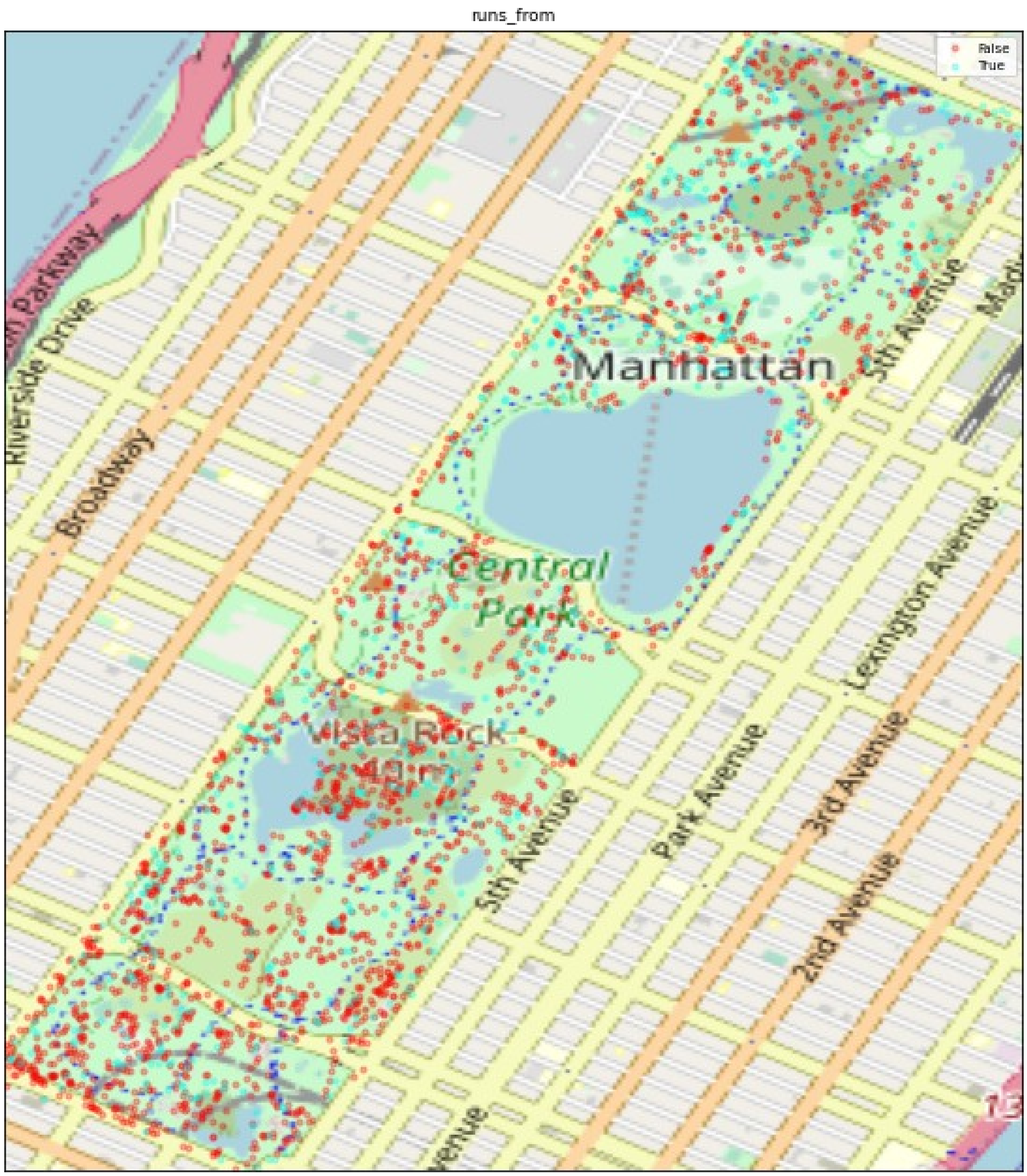


approaches


False
True
Riverside Drive
Broadway
Manhattan
Central Park
Vista Rock
5th Avenue
5th Avenue
Lexington Avenue
Park Avenue
3rd Avenue
2nd Avenue

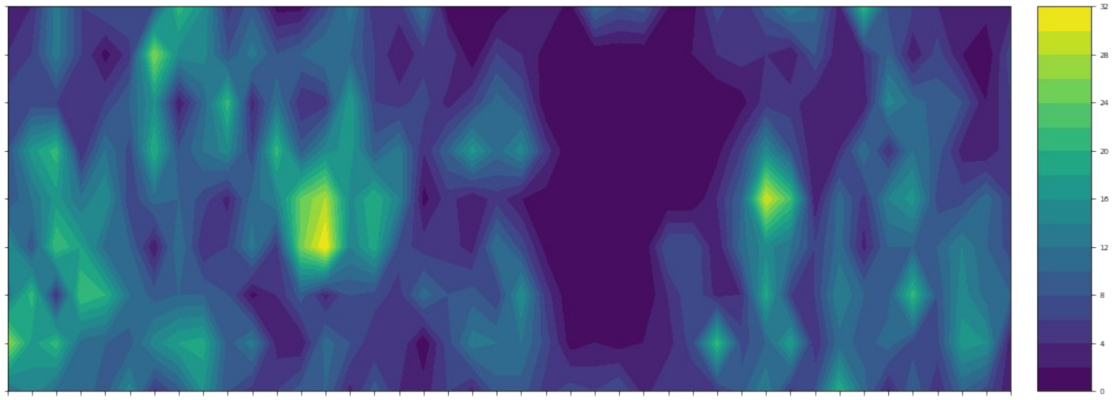
indifferent





We also created a heat-map from the same data, by not using the coordinate information but hectare information, which can be seen in Fig. \ref{heatmap}. Despite being stripped of the identifying information of the geography of Central Park, it lets us guess at not only the the shape of the park but also the altitude even though the altitude information is not given at all in the data. The presence of the squirrels forms a kind of topographical map of the park. Especially, from the top-right of Fig. \ref{colormap}, it can be seen that squirrels with black primary color creates contours of the map.

```
heatmap(nyc_squirrels, save_fig=False)
```



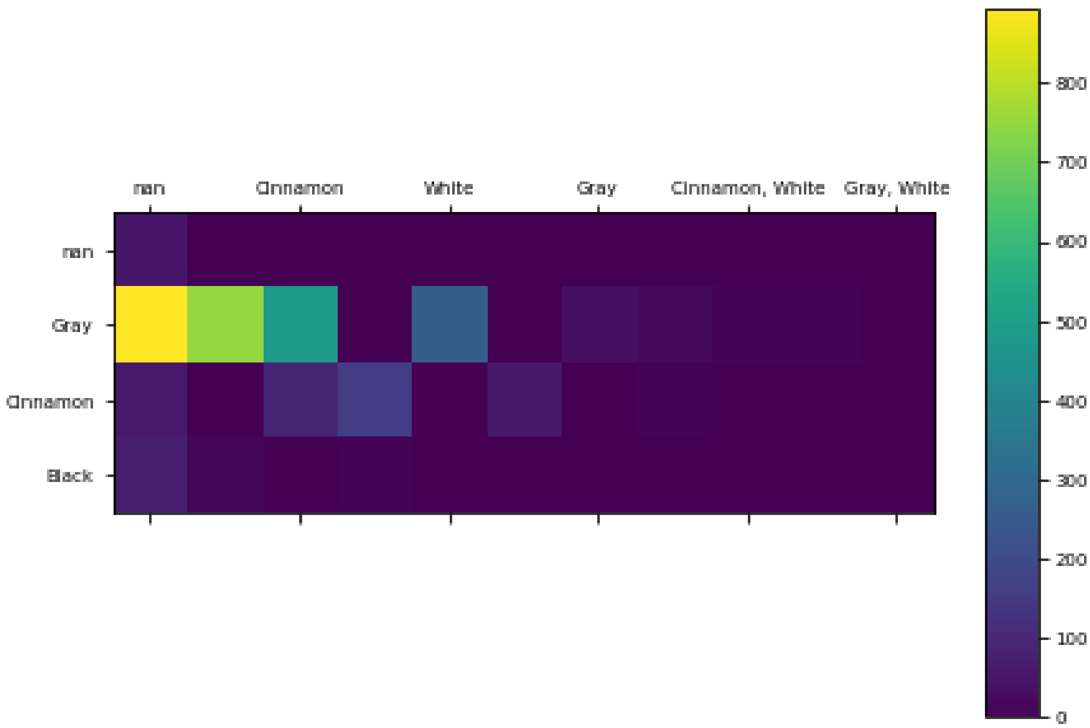
Fur Colors

Next, we took a closer look at the different colors of the squirrels. Each squirrel has one of three primary colors (Gray, Cinnamon, Black) or None if the recorder failed to identify the color. Each squirrel can then have any combination of those 3 colors or White as a highlight color (including having no highlight color at all). The split of highlight colors over the three primary colors is shown. None was omitted as the only highlight color recorded was None. Even after taking into consideration that primary and highlight color can not match, it is evident that the highlight colors follow different distribution depending on the primary color. Cinnamon squirrels show the highest variety in highlight color, with white, gray and cinnamon all being prevalent. Cinnamon is especially the only primary color with a significant level of tricolored squirrels. Black squirrels are primarily mono-colored with some few individuals having cinnamon or gray highlights.

```
color_combo = nyc_squirrels[['primary_fur_color',
'highlight_fur_color']]
color_combo['num_primary'] = [numColor(item) for item in
color_combo['primary_fur_color']]
color_combo['num_highlight'] = [numColor(item) for item in
color_combo['highlight_fur_color']]

primaryHighlightCombination = pd.DataFrame(\
                                            np.zeros([4,11]),\

index=color_combo.primary_fur_color.unique(),\

columns=color_combo.highlight_fur_color.unique())

for i in range(color_combo.shape[0]):
  item = color_combo.iloc[i]
  prim = item['primary_fur_color']
  high = item['highlight_fur_color']
  primaryHighlightCombination.at[prim, high] += 1


fig = plt.figure()
```

```python
ax = fig.add_subplot(111)
cax = ax.matshow(primaryHighlightCombination)
fig.colorbar(cax)

ax.set_xticklabels([''] 
+color_combo.highlight_fur_color.unique().tolist())
ax.set_yticklabels([''] 
+color_combo.primary_fur_color.unique().tolist())

plt.show()
plt.savefig("fig/fur_relations.pdf",bbox_inches="tight")
```

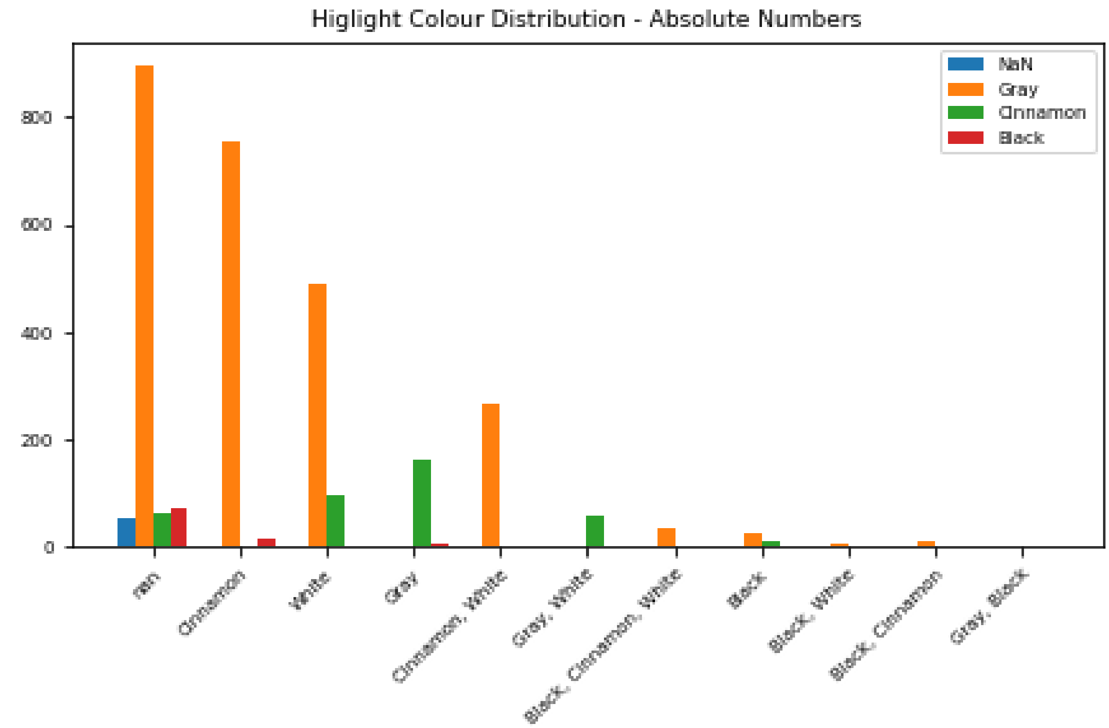
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: 
SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: 
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: 
SettingWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: 
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing 
imports until
```

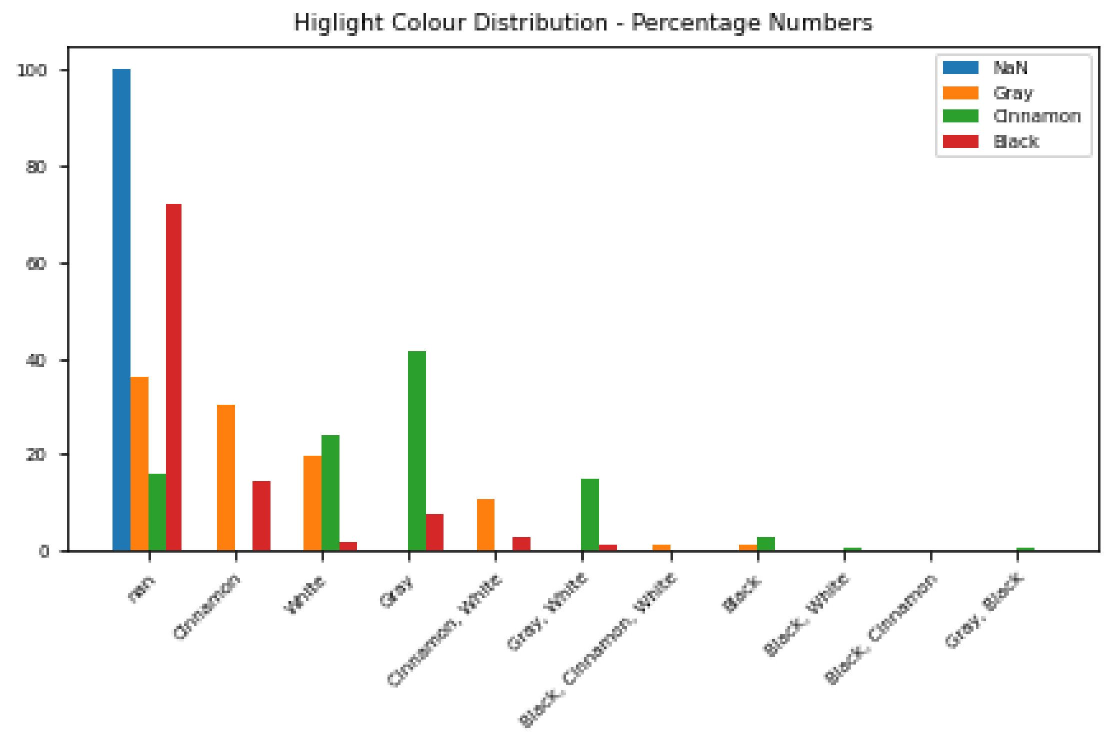



```
<Figure size 432x288 with 0 Axes>
```

```python
#Total Distribution of Highlight Colours
fig, ax = plt.subplots()
x = np.arange(len(primaryHighlightCombination.columns))
ax.bar(x-0.3, primaryHighlightCombination.loc[np.nan], width=0.2,
label="NaN")
ax.bar(x-0.1, primaryHighlightCombination.loc['Gray'], width=0.2,
label="Gray")
ax.bar(x+0.1, primaryHighlightCombination.loc['Cinnamon'], width=0.2,
label="Cinnamon")
ax.bar(x+0.3, primaryHighlightCombination.loc['Black'], width=0.2,
label="Black")

ax.set_title("Higlight Colour Distribution - Absolute Numbers")
ax.legend()
ax.set_xticks(np.arange(len(primaryHighlightCombination.columns)))
ax.set_xticklabels(primaryHighlightCombination.columns)
plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
rotation_mode="anchor");
plt.savefig("fig/highlight_dist_abs.pdf",bbox_inches="tight")

fig, ax = plt.subplots()
nan = primaryHighlightCombination.loc[np.nan].sum()/100
gray = primaryHighlightCombination.loc['Gray'].sum()/100
cinnamon = primaryHighlightCombination.loc['Cinnamon'].sum()/100
black = primaryHighlightCombination.loc['Black'].sum()/100
```

```
ax.bar(x-0.3, primaryHighlightCombination.loc[np.nan]/nan, width=0.2,
label="NaN")
ax.bar(x-0.1, primaryHighlightCombination.loc['Gray']/gray, width=0.2,
label="Gray")
ax.bar(x+0.1, primaryHighlightCombination.loc['Cinnamon']/cinnamon,
width=0.2, label="Cinnamon")
ax.bar(x+0.3, primaryHighlightCombination.loc['Black']/black,
width=0.2, label="Black")

ax.set_title("Higlight Colour Distribution - Percentage Numbers")
ax.legend()
ax.set_xticks(np.arange(len(primaryHighlightCombination.columns)))
ax.set_xticklabels(primaryHighlightCombination.columns)
plt.setp(ax.get_xticklabels(), rotation=45,
ha="right",rotation_mode="anchor");
plt.savefig("fig/highlight_dist_per.pdf",bbox_inches="tight")
```





Behavioural Correlation

"Behaviour and Time", and "Behaviour and Color" are investigated.

```
def behaviourUpdate(frame,object,keyword):
  #Reads out all behaviours for a given squirrel and adds them to a
dataframe under the keyword
  for column in frame.columns:
    if(object[column]):
      frame[column][keyword] += 1

def update(frame, object, keyword):
  #Checks for keyword presence then reads behaviour
  if(object[keyword]):
    behaviourUpdate(frame, object, keyword)

# Time
color_time = nyc_squirrels[['primary_fur_color', 'slot']]

colorTimeCombo= pd.DataFrame(\
                             np.zeros([4,2]),\

index=color_time.primary_fur_color.unique(),\
                             columns=color_time.slot.unique())

for i in range(color_time.shape[0]):
  item = color_time.iloc[i]
```

```
  prim = item['primary_fur_color']
  time = item['slot']
  colorTimeCombo.at[prim, time] += 1
colorTimeCombo.loc[np.nan] = colorTimeCombo.loc[np.nan]/nan
colorTimeCombo.loc["Gray"] = colorTimeCombo.loc["Gray"]/gray
colorTimeCombo.loc["Cinnamon"] =
colorTimeCombo.loc["Cinnamon"]/cinnamon
colorTimeCombo.loc["Black"] = colorTimeCombo.loc["Black"]/black

colorTimeCombo
```

```
                 PM         AM
NaN       47.272727  52.727273
Gray      56.045289  43.954711
Cinnamon  54.081633  45.918367
Black     50.485437  49.514563
```

```
# P(Column | Row) but for Behaviour and Color
color_behaviour_relation = pd.DataFrame(\
                                        np.zeros([4,13]),\
                                        index=[np.nan, "Gray",
"Cinnamon", "Black"],\
                                        columns=["running", "chasing",
"climbing", "eating", "foraging", "kuks", "quaas", "moans",
"tail_flags", "tail_twitches", "approaches", "indifferent",
"runs_from"])

for i in range(len(nyc_squirrels.index)):
  row = nyc_squirrels.iloc[i]
  keyword = row["primary_fur_color"]
  behaviourUpdate(color_behaviour_relation, row, keyword)

#Normalize to percentages
for col in color_behaviour_relation.columns:
  color_behaviour_relation[col][np.nan] =\
   round(color_behaviour_relation[col][np.nan]/nan,1)
  color_behaviour_relation[col]["Gray"] =\
   round(color_behaviour_relation[col]["Gray"]/gray,1)
  color_behaviour_relation[col]["Cinnamon"] =\
   round(color_behaviour_relation[col]["Cinnamon"]/cinnamon,1)
  color_behaviour_relation[col]["Black"] =\
   round(color_behaviour_relation[col]["Black"]/black,1)

color_behaviour_relation
```

```
          running  chasing  climbing  ...  approaches  indifferent
runs_from
NaN           7.3      7.3      32.7  ...         3.6         18.2
14.5
Gray         24.1      9.6      21.6  ...         5.1         49.3
```

```
22.3
Cinnamon     26.3      7.7      20.9   ...         11.2           46.2
22.2
Black        25.2      6.8      24.3   ...          5.8           42.7
31.1

[4 rows x 13 columns]
```

Behavioural Correlation - Internal

We investigated the correlation between different behaviours of squirrels. It should be noted that due to being nominal data, the information given is in the form of conditional probability, not covariance. While we can see what appears like strong relations especially in the case of "foraging" and "indifferent", these should be treated with caution as they each make up about 50% of all recorded squirrels. Consequently the strong relation can actually be seen in those examples where the odds are disproportionately low. This is the case for intuitively mutually exclusive behaviours like indifferent-approaches-runs_from or foraging-climbing-chasing. Kukking and quaaing also predict against indifferent. As these are both danger signalling noises this is consistent with expectations. Moaning should be disregarded due to only showing up 3 times across all sightings.

This is P(Column | Row) not Covariance!

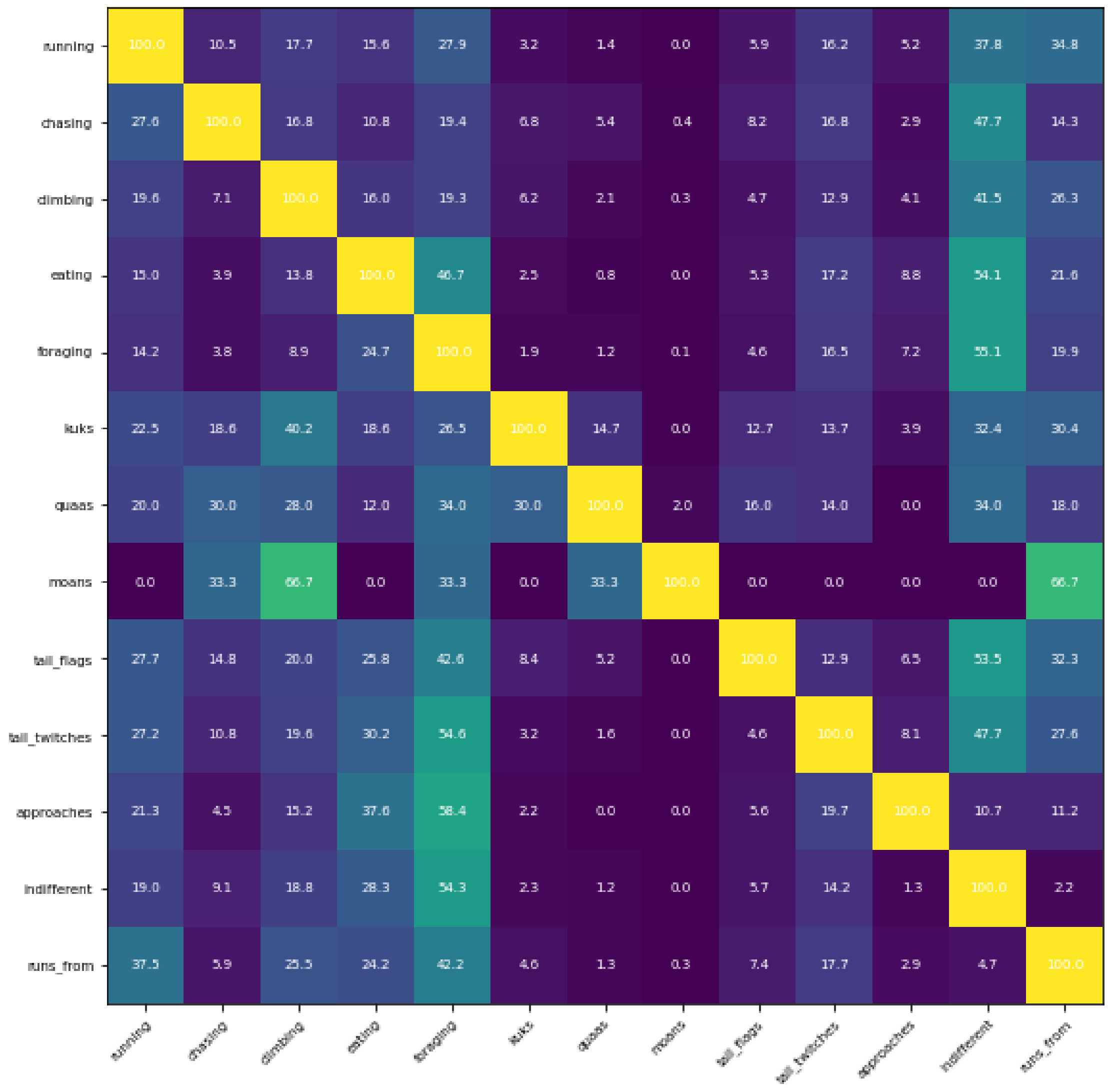
That's why we get asymmetry.

```
behaviours = nyc_squirrels[["running", "chasing", "climbing",
"eating", "foraging", "kuks", "quaas", "moans", "tail_flags",
"tail_twitches", "approaches", "indifferent", "runs_from"]]

behaviour_relation = pd.DataFrame(np.zeros([behaviours.columns.size,
behaviours.columns.size]), index=behaviours.columns,
columns=behaviours.columns)

for index, row in behaviours.iterrows():
  for column in behaviour_relation.columns:
    update(behaviour_relation, row, column)

for row in behaviour_relation.index:
  ref_val = behaviour_relation[row][row]
  print(str(row) + " Total: " + str(ref_val))
  for col in behaviour_relation.columns:
    behaviour_relation[col][row] = round((behaviour_relation[col]
[row]/ref_val)*100, 1)
```

```
running Total: 730.0
chasing Total: 279.0
climbing Total: 658.0
eating Total: 760.0
foraging Total: 1435.0
kuks Total: 102.0
```

```
quaas Total: 50.0
moans Total: 3.0
tail_flags Total: 155.0
tail_twitches Total: 434.0
approaches Total: 178.0
indifferent Total: 1454.0
runs_from Total: 678.0
```

```
plot_behaviourmap(nyc_squirrels, behaviours, behaviour_relation,
save_fig=False)
```

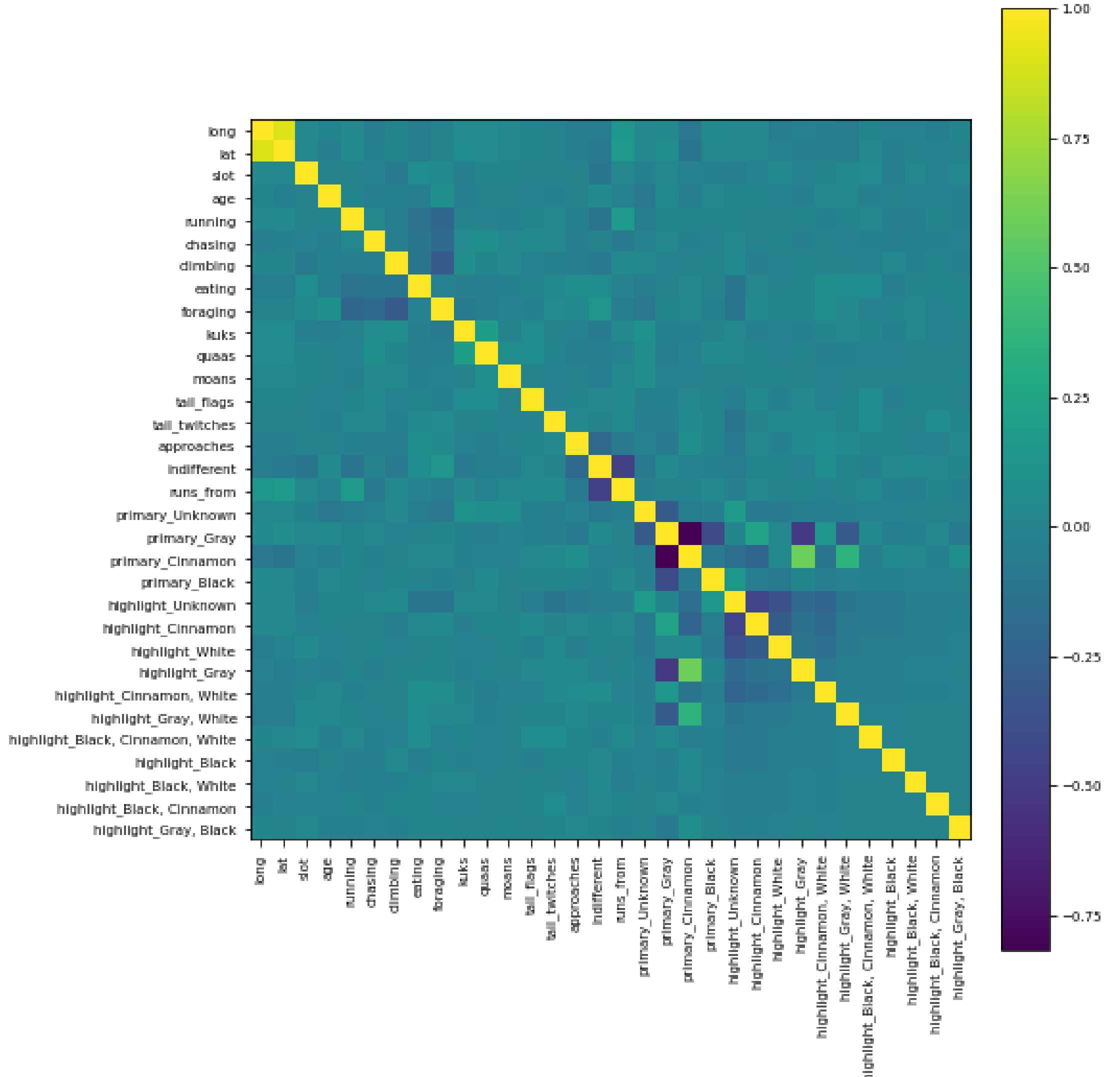


Numerating the data

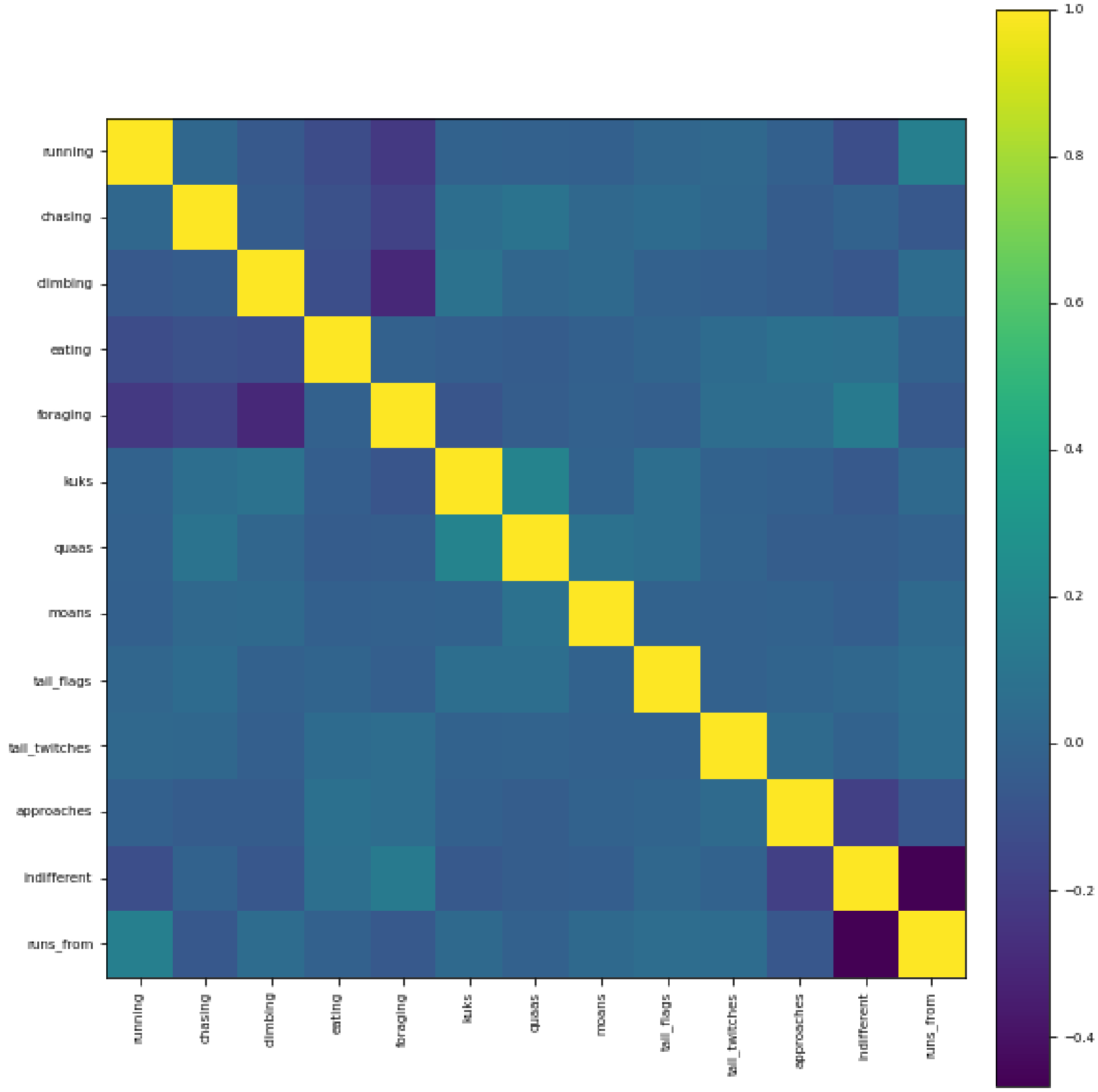
Here, we normalized the data column-wise with min-max normalization after numerating.

```
nyc_squirrels = dataset_numerate(nyc_squirrels)
normalized_nyc_squirrels = nyc_squirrels
```

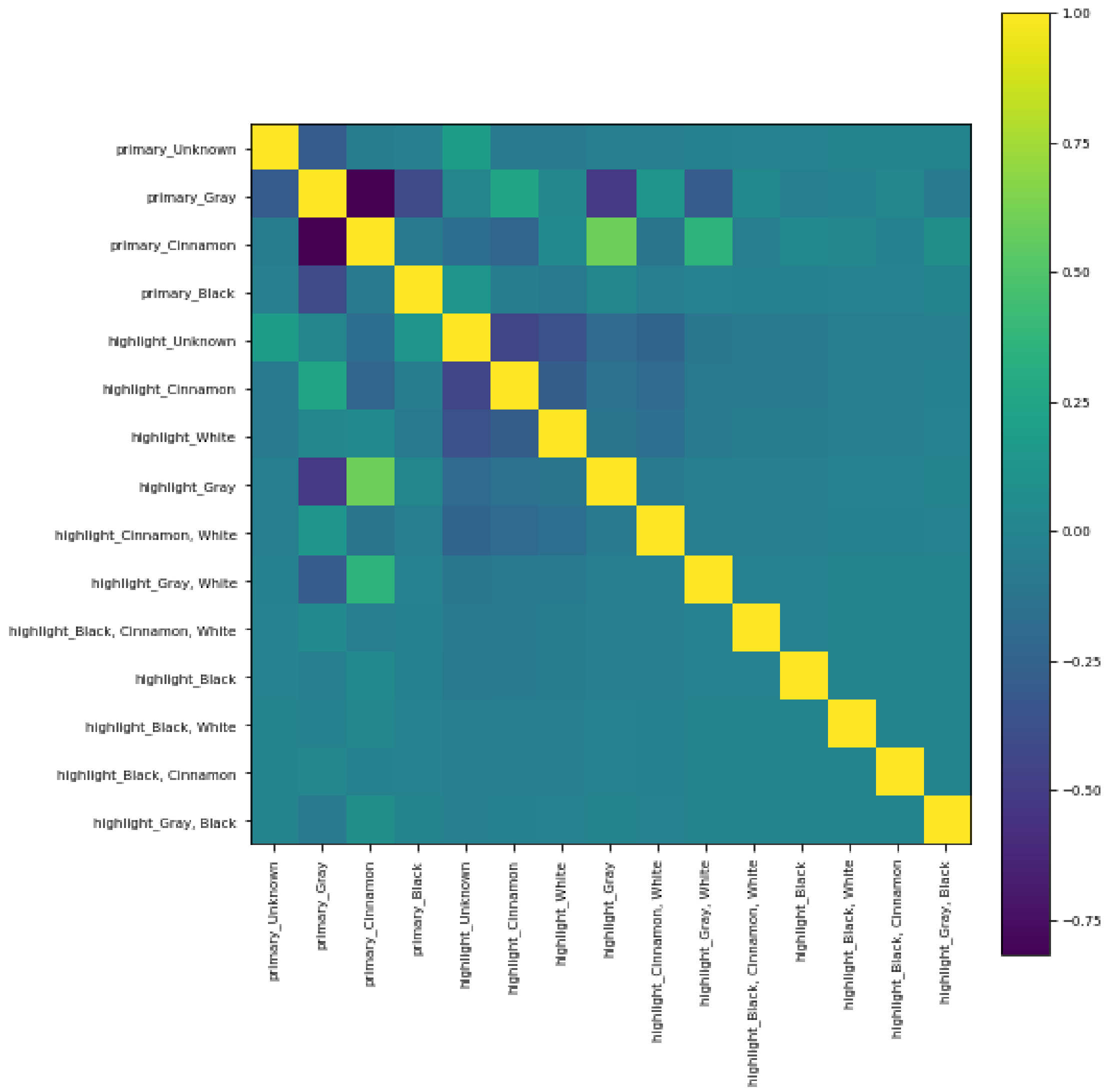
```python
# Column-wise min-max normalization
for x in nyc_squirrels.columns.unique():
  normalized_nyc_squirrels[x]=(nyc_squirrels[x]-
nyc_squirrels[x].min())/(nyc_squirrels[x].max()-
nyc_squirrels[x].min())

nyc_squirrels = normalized_nyc_squirrels

behaviours = ["running", "chasing", "climbing", "eating", "foraging",
"kuks",
          "quaas", "moans", "tail_flags", "tail_twitches",
"approaches",
          "indifferent", "runs_from"]

colours = ['primary_Unknown', 'primary_Gray', 'primary_Cinnamon',
'primary_Black',
       'highlight_Unknown', 'highlight_Cinnamon', 'highlight_White',
       'highlight_Gray', 'highlight_Cinnamon, White', 'highlight_Gray,
White',
       'highlight_Black, Cinnamon, White', 'highlight_Black',
       'highlight_Black, White', 'highlight_Black, Cinnamon',
       'highlight_Gray, Black']

plot_correlation(nyc_squirrels, nyc_squirrels.columns, "all",
save_fig=False)
plot_correlation(nyc_squirrels, behaviours, "behaviours",
save_fig=False)
plot_correlation(nyc_squirrels, colours, "colours", save_fig=False)
```



long
lat
slot
age
running
chasing
climbing
eating
foraging
kuks
quaas
moans
tail_flags
tail_twitches
approaches
indifferent
runs_from
primary_Unknown
primary_Gray
primary_Cinnamon
primary_Black
highlight_Unknown
highlight_Cinnamon
highlight_White
highlight_Gray
highlight_Cinnamon, White
highlight_Gray, White
highlight_Black, Cinnamon, White
highlight_Black
highlight_Black, White
highlight_Black, Cinnamon
highlight_Gray, Black
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75



running
chasing
climbing
eating
foraging
kuks
quaas
moans
tail_flags
tail_twitches
approaches
indifferent
runs_from
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

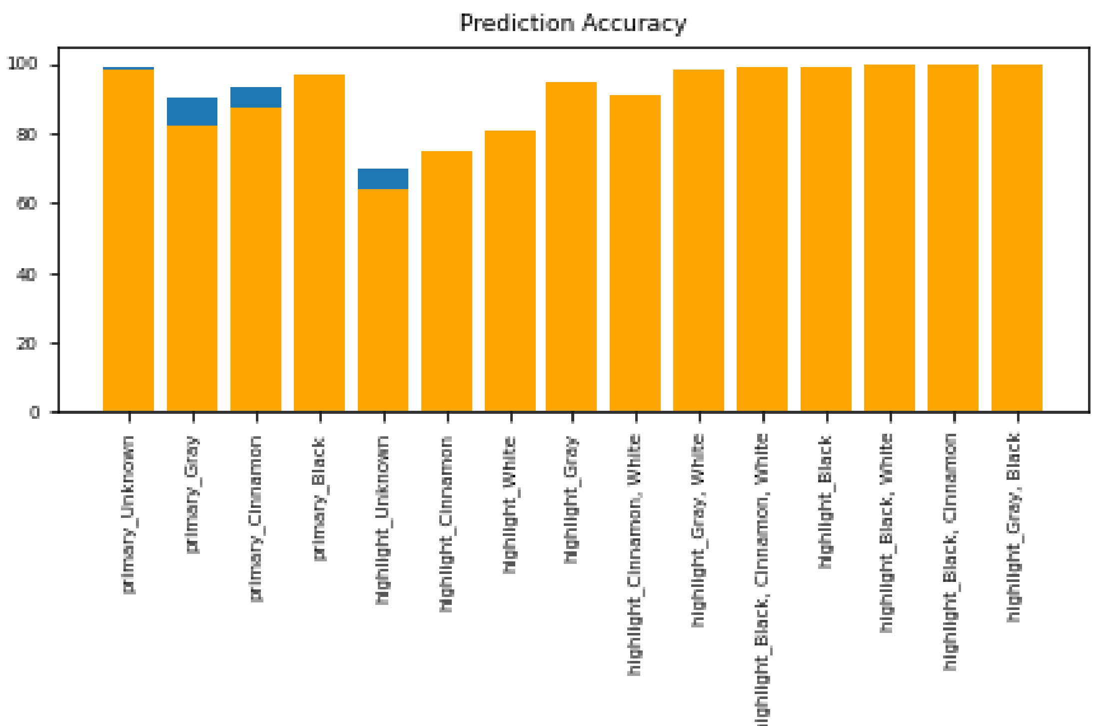


Logistic Regression

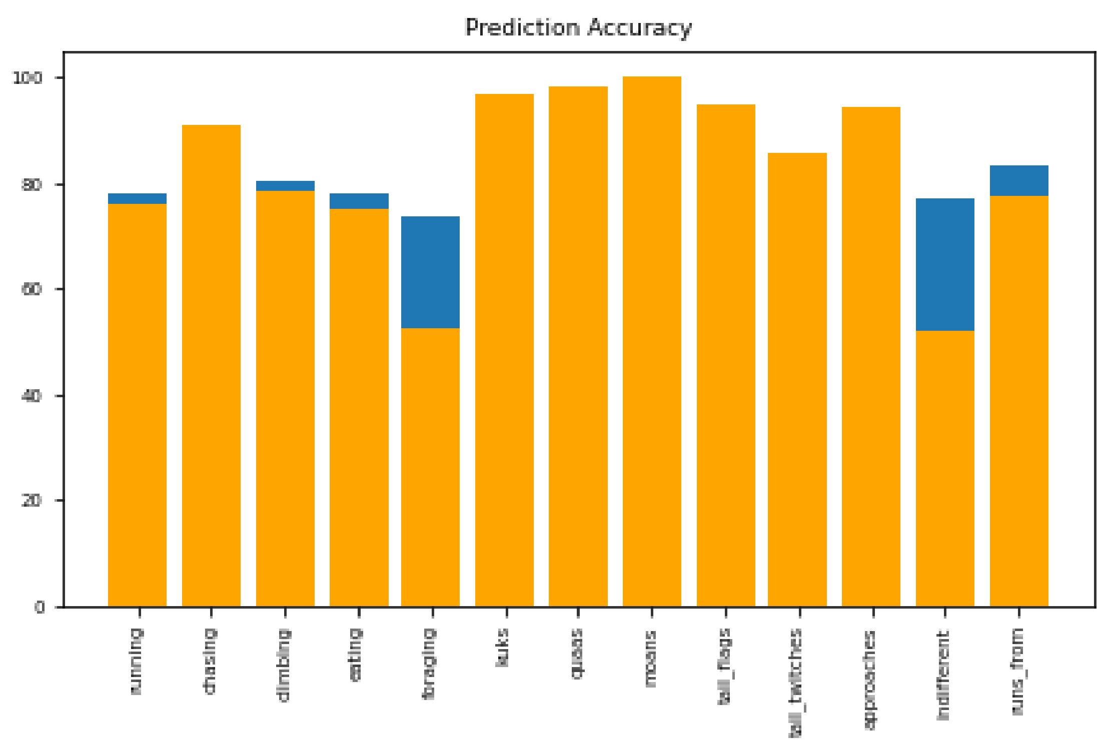
In addition to the above analysis, we created a logistic regression model of different squirrel qualities. We used a Training/Test split of 75/25. We see that a logistic regression is able to achieve some success in predicting the primary color from the highlight colors and significant success in determining whether a squirrel was eating and whether it was indifferent to humans from its other behaviours.

```
def make_predictions(nyc_squirrels, prediction_of = "primary_Gray",
train_test_ratio = 0.75, mod="LogisticRegression"):

  predictors, label, columns = x_y_pairs(nyc_squirrels, prediction_of)
  train_X, train_Y, test_X, test_Y = split_data(predictors, label,
train_test_ratio)
  model = fit(train_X, train_Y, mod)
  acc = test_accuracy(model, test_X, test_Y, mod)
  coef_dic = dict(zip(columns, model.coef_.reshape(-1,1)))
```

```
  return model, coef_dic, acc

colours = ['primary_Unknown', 'primary_Gray', 'primary_Cinnamon',
'primary_Black',
       'highlight_Unknown', 'highlight_Cinnamon', 'highlight_White',
       'highlight_Gray', 'highlight_Cinnamon, White', 'highlight_Gray,
White',
       'highlight_Black, Cinnamon, White', 'highlight_Black',
       'highlight_Black, White', 'highlight_Black, Cinnamon',
       'highlight_Gray, Black']

colour_acc = []
count_common = []
for colour in colours:
  _, _, acc = make_predictions(nyc_squirrels, colour)
  colour_acc.append(acc)

count_common.append(pd.DataFrame(nyc_squirrels[colour].value_counts())
.reset_index()[colour][0])
count_common = 100*np.array(count_common)/(nyc_squirrels.shape[0])
colours_acc = dict(zip(colours, colour_acc))
colours_count = dict(zip(colours, count_common.tolist()))
dict_plot(colours_acc, colours_count, name = "colours_acc",
save_fig=False)
```



```
behaviours = ['running', 'chasing', 'climbing',
       'eating', 'foraging', 'kuks', 'quaas', 'moans', 'tail_flags',
       'tail_twitches', 'approaches', 'indifferent', 'runs_from']

behaviour_acc = []
count_common = []
for behaviour in behaviours:
  _, _, acc = make_predictions(nyc_squirrels, behaviour)
  behaviour_acc.append(acc)

count_common.append(pd.DataFrame(nyc_squirrels[behaviour].value_counts
()).reset_index()[behaviour][0])
count_common = 100*np.array(count_common)/(nyc_squirrels.shape[0])
behaviours_acc = dict(zip(behaviours, behaviour_acc))
behaviours_count = dict(zip(behaviours, count_common.tolist()))
dict_plot(behaviours_acc, behaviours_count, "behaviours_acc",
save_fig=False)
```



Mean Difference and Normalized Mean Difference Calculation

We also used the Themis-ML, developed by \cite{bantilan2017} to calculate mean difference scores for color and age with regard to foraging. We found that black squirrels were more likely to be foraging while juveniles were less likely. Perhaps black squirrels are more curious and less wary to be humans and consequently approach them, while juvenile squirrels are either cared for by their parents or simply more wary of the unknown.

```
!pip install themis_ml
```

```
Requirement already satisfied: themis_ml in
/usr/local/lib/python3.7/dist-packages (0.0.4)
Requirement already satisfied: numpy>=1.9.0 in
/usr/local/lib/python3.7/dist-packages (from themis_ml) (1.19.5)
Requirement already satisfied: pandas>=0.22.0 in
/usr/local/lib/python3.7/dist-packages (from themis_ml) (1.3.5)
Requirement already satisfied: pathlib2 in
/usr/local/lib/python3.7/dist-packages (from themis_ml) (2.3.6)
Requirement already satisfied: scikit-learn>=0.19.1 in
/usr/local/lib/python3.7/dist-packages (from themis_ml) (1.0.2)
Requirement already satisfied: scipy>=0.19.1 in
/usr/local/lib/python3.7/dist-packages (from themis_ml) (1.4.1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/usr/local/lib/python3.7/dist-packages (from pandas>=0.22.0-
>themis_ml) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in
/usr/local/lib/python3.7/dist-packages (from pandas>=0.22.0-
>themis_ml) (2018.9)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3-
>pandas>=0.22.0->themis_ml) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.19.1-
>themis_ml) (3.1.0)
Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.19.1-
>themis_ml) (1.1.0)
```

```python
from themis_ml.metrics import mean_difference,
normalized_mean_difference

sensitive_attributes = ["primary_Gray", "primary_Cinnamon",
"primary_Black", "age"]
for sensitive in sensitive_attributes:
  if sensitive == "age":
    attribute = nyc_squirrels[sensitive].replace(0.5, 0)
  else:
    attribute = nyc_squirrels[sensitive]
  foraging = nyc_squirrels["foraging"]

  print("\nMean difference scores:")
  print("protected class = "+str(sensitive)+": ",
str(mean_difference(foraging, attribute)[0]))
  print("Normalized mean difference scores:")
  print("protected class = "+str(sensitive)+": ",
str(normalized_mean_difference(foraging, attribute)[0]))
```

```
Mean difference scores:
protected class = primary_Gray:  -0.020184538470021662
```

```
Normalized mean difference scores:
protected class = primary_Gray:  -0.031433478360430464

Mean difference scores:
protected class = primary_Cinnamon:  -0.04959412499321286
Normalized mean difference scores:
protected class = primary_Cinnamon:  -0.09092832254853174

Mean difference scores:
protected class = primary_Black:  0.059236600611783485
Normalized mean difference scores:
protected class = primary_Black:  0.12053719427624235

Mean difference scores:
protected class = age:  -0.13967255485947072
Normalized mean difference scores:
protected class = age:  -0.22586846402967306
```