Задача 6

Набор данных:  <https://www.kaggle.com/new-york-city/ny-2015-street-tree-census-tree-data?select=2015-street-tree-census-tree-data.csv>

Задачи:

1. выделить районы, в которых состояние деревьев аномально хорошее или плохое;
2. выделить наилучший район по состоянию деревьев;
3. выделить виды деревьев (или более общие элементы классификации), обладающие наилучшими показателями «здоровья»

Решение:

1. Считываем набор данных и обрабатываем его. Смотри приложение 1, строки из Приложения 1: 1-157.
2. Опишем каждый признак. Смотри пункт 2 из задачи 5.
3. Задача 1. Выделить районы, в которых состояние деревьев аномально хорошее или плохое.
   1. Создадим новый признак “problems”, состоящий из суммы положительных ответов о проблемах деревьев. Смотри рисунок 1.
   2. Построим пару графиков для поиска аномальных состояний деревьев. Смотри рисунок 1.

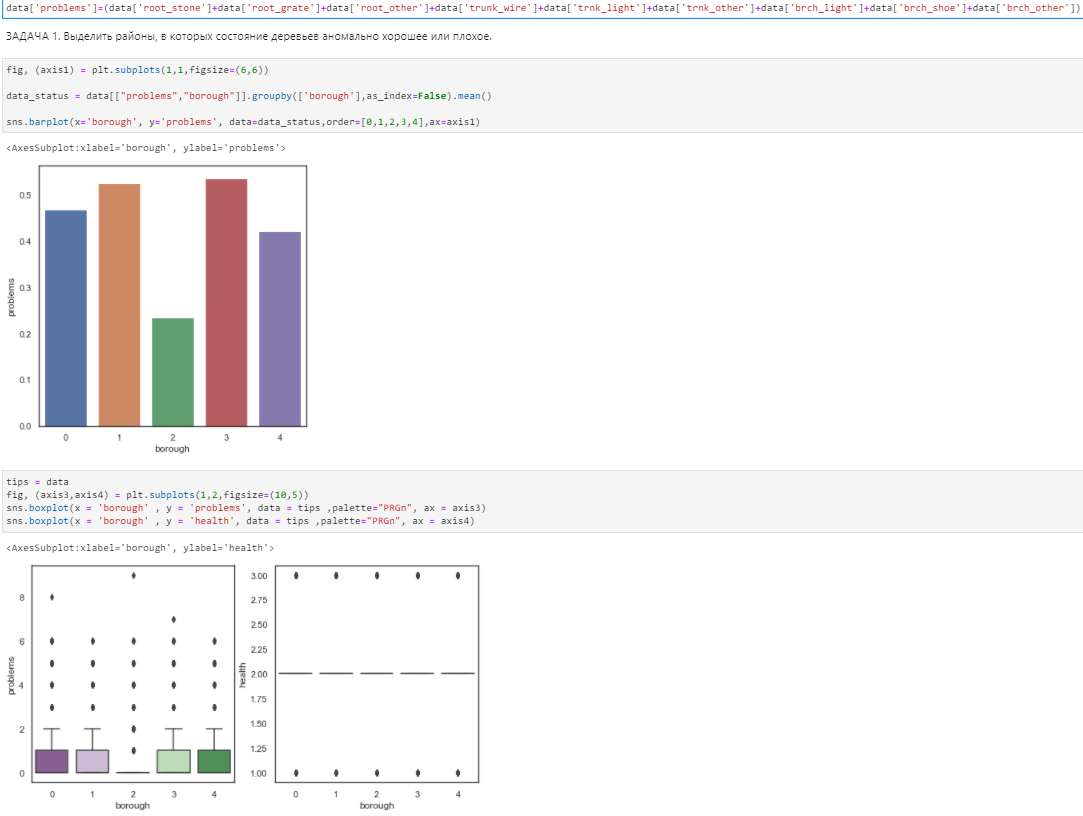


Рисунок 1. Создание нового признака [‘problems’] и 3-ёх графиков для поиска аномальных значений.

Проанализировав графики можно прийти к выводу, что деревья в районе под номером 2 - Staten Island имеют аномально хорошее состояние.

1. ЗАДАЧА 2. Выделить наилучший район по состоянию деревьев.

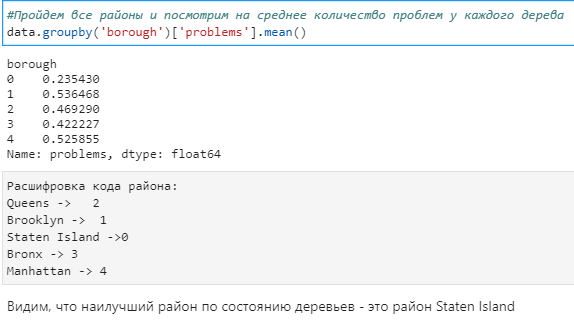


Рисунок 2. Поиск района с наименьшим количеством проблем у деревьев.

Из данных с рисунка 2 видим, что район с наилучшим состоянием деревьев – это Staten Island.

1. ЗАДАЧА 3. Выделить виды деревьев (или более общие элементы классификации), обладающие наилучшими показателями «здоровья»

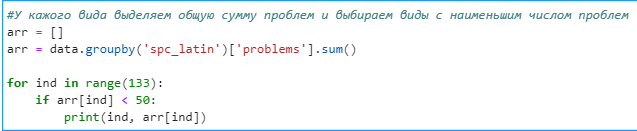


Рисунок 3. Суммируем число проблем у каждого вида дерева. Элемент классифицируется по его названию на Латыни.

Проанализировав результат фрагмента кода из рисунка 3 имеем, что виды деревья с наилучшим состоянием здоровья – это деревья:

1. Tilia cordata
2. Cotinus coggygria
3. Quercus alba
4. Ulmus pumila
5. Acer buergerianum
6. Acer nigrum
7. Pinus resinosa
8. Quercus rubra
9. Cornus florida
10. Carpinus betulus
11. Cedrus atlantica
12. Quercus velutina
13. Liquidambar styraciflua
14. Prunus virginiana
15. Picea
16. Aesculus glabra
17. Fagus grandifolia
18. Cornus mas
19. Magnolia grandiflora
20. Carya glabra
21. Acer tataricum
22. Syringa reticulata
23. Carpinus caroliniana
24. Chamaecyparis pisifera
25. Carpinus japonica
26. Fraxinus pennsylvanica
27. Pyrus calleryana
28. Crataegus
29. Acer negundo
30. Tsuga canadensis
31. Parrotia persica
32. Acer ginnala
33. Pinus

Вывод:

Основываясь на результатах из пунктов 3,4,5 мы приходим к выводу, что район с аномально хорошим и наилучшим состоянием деревьев – это район Staten Island; виды деревьев с наилучшими показателями здоровья - Смотри вывод из пункта 5.

Приложение 1:

1. import pandas as pd
2. import numpy as np
3. import matplotlib.pyplot as plt # библиотека для визуализации
4. import seaborn as sns
5. sns.set(style="white", color\_codes=True)
6. %matplotlib inline
7. import warnings
8. warnings.filterwarnings('ignore')
9. data = pd.read\_csv('2015-street-tree-census-tree-data.csv')
10. import math
11. arr = ['block\_id','tree\_dbh','stump\_diam','postcode','community board','cncldist','st\_assem','st\_senate','latitude','longitude','x\_sp','y\_sp']
12. for column in arr:
13. data[column] = (data[column] - data[column].mean())/(math.sqrt(data[column].var()))
15. data['state'] = np.where(data['state'] == 'New York', 1, 0)
16. #Расположено ли дерево вдоль бордюра или со смещением от него
17. data['curb\_loc'] = np.where(data['curb\_loc'] == 'OnCurb', 1, 0)
18. #Повреждение тратуара в непосредственной близости от дерева
19. data['sidewalk'] = np.where(data['sidewalk'] == 'Damage', 1, 0)
20. #Указывает на наличие проблемы с корнями, вызванной брусчаткой в ложе дерева
21. data['root\_stone'] = np.where(data['root\_stone'] == 'Yes', 1, 0)
22. #Указывает на наличие проблемы с корнями, вызванной металлическими решетками в ложе дерева
23. data['root\_grate'] = np.where(data['root\_grate'] == 'Yes', 1, 0)
24. #Указывает на наличие других корневых проблем
25. data['root\_other'] = np.where(data['root\_other'] == 'Yes', 1, 0)
26. #Указывает на наличие проблемы со стволом, вызванной проволокой или веревкой, обмотанной вокруг ствола
27. data['trunk\_wire'] = np.where(data['trunk\_wire'] == 'Yes', 1, 0)
28. #Указывает на наличие проблем со стволом, вызванных установленным на дереве освещением
29. data['trnk\_light'] = np.where(data['trnk\_light'] == 'Yes', 1, 0)
30. #Указывает на наличие других проблем с магистралью
31. data['trnk\_other'] = np.where(data['trnk\_other'] == 'Yes', 1, 0)
32. #Указывает на наличие проблемы с ветвями, вызванной светильниками (обычно струнными) или проводами в ветвях
33. data['brch\_light'] = np.where(data['brch\_light'] == 'Yes', 1, 0)
34. #Указывает на наличие проблемы с ветками, вызванной кроссовками в ветках
35. data['brch\_shoe'] = np.where(data['brch\_shoe'] == 'Yes', 1, 0)
36. #Указывает на наличие других проблем с ветвями
37. data['brch\_other'] = np.where(data['brch\_other'] == 'Yes', 1, 0)
38. #Город, полученный из почтового индекса
39. my\_set = set(data.zip\_city)
40. i = 0
41. for item in my\_set:
42. data['zip\_city'] = data['zip\_city'].replace(item, i)
43. i = i + 1
44. #Это название NTA, соответствующее области табуляции района по переписи населения США 2010 года, в которую попадает точка дерева
45. my\_set = set(data.nta\_name)
46. i = 0
47. for item in my\_set:
48. data['nta\_name'] = data['nta\_name'].replace(item, i)
49. i = i + 1
50. #Это код NTA, соответствующий району, в который попадает точка дерева, согласно переписи населения США 2010 года.
51. my\_set = set(data.nta)
52. i = 0
53. for item in my\_set:
54. data['nta'] = data['nta'].replace(item, i)
55. i = i + 1
56. #Название района, в котором находится точка произрастания деревьев
57. my\_set = set(data.borough)
58. i = 0
59. for item in my\_set:
60. data['borough'] = data['borough'].replace(item, i)
61. i = i + 1
62. #Город, полученный из почтового индекса. Часто (но не всегда) это то же самое, что и район.
63. my\_set = set(data.postcode)
64. i = 0
65. for item in my\_set:
66. data['postcode'] = data['postcode'].replace(item, i)
67. i = i + 1
68. #Указывает, присутствует ли ограждение, и считает ли пользователь, что это полезное или вредное ограждение. Не регистрируется для мертвых деревьев и пней.
69. my\_set = set(data.guards)
70. i = 0
71. for item in my\_set:
72. data['guards'] = data['guards'].replace(item, i)
73. i = i + 1
74. #Это поле описывает категорию пользователя, который собрал данные этой точки дерева.
75. my\_set = set(data.user\_type)
76. i = 0
77. for item in my\_set:
78. data['user\_type'] = data['user\_type'].replace(item, i)
79. i = i + 1
81. #Указывает количество уникальных признаков бережного отношения, отмеченных для данного дерева. Не регистрируется для пней или мертвых деревьев.
82. my\_set = set(data.steward)
83. i = 0
84. for item in my\_set:
85. data['steward'] = data['steward'].replace(item, i)
86. i = i + 1
87. #Общее название вида, например, "красный клен"
88. my\_set = set(data.spc\_common)
89. i = 0
90. for item in my\_set:
91. data['spc\_common'] = data['spc\_common'].replace(item, i)
92. i = i + 1
94. #Научное название вида, например, "Acer rubrum".
95. my\_set = set(data.spc\_latin)
96. i = 0
97. for item in my\_set:
98. data['spc\_latin'] = data['spc\_latin'].replace(item, i)
99. i = i + 1
100. #Здоровье деревьев
101. my\_set = set(data.health)
102. i = 0
103. for item in my\_set:
104. data['health'] = data['health'].replace(item, i)
105. i = i + 1
107. data.loc[data.health == 'NaN', 'health'] = 0
108. data.loc[data.health == 0, 'health'] = 3
110. #Указывает, является ли дерево живым, стоящим мертвым или пнем.
111. data.loc[data.status == 'Alive', 'status'] = 2
112. data.loc[data.status == 'Stump', 'status'] = 1
113. data.loc[data.status == 'Dead', 'status'] = 0
114. data.loc[data.spc\_latin == 'NaN', 'spc\_latin'] = 0
115. data.loc[data.spc\_common == 'NaN', 'spc\_common'] = 0
116. data.loc[data.steward == 'NaN', 'steward'] = 0
117. data.loc[data.guards == 'NaN', 'guards'] = 0
118. data.loc[data.sidewalk == 'NaN', 'sidewalk'] = 0
119. #исключим столбец с уникальными значениями: tree\_id
120. data = data.drop('tree\_id', axis = 1)
121. #исключим столбец, включающий результаты других столбцов
122. data = data.drop('problems', axis = 1)
123. #исключим столбец с индивидуальными параметрами
124. data = data.drop('created\_at', axis = 1)
125. #исключим столбец с индивидуальными параметрами
126. data = data.drop('address', axis = 1)
127. #исключим столбцы с признаками, которые отсутствуют в документации
128. data = data.drop('council district', axis = 1)
129. data = data.drop('census tract', axis = 1)
130. data = data.drop('bin', axis = 1)
131. data = data.drop('bbl', axis = 1)
132. #Создадим новый признак - problems {тип: число}, состоит из суммы положительных ответов о проблемах корней
133. #Чем больше значение problems - тем состояние дерева хуже
134. data['problems']=(data['root\_stone']+data['root\_grate']+data['root\_other']+data['trunk\_wire']+data['trnk\_light']+data['trnk\_other']+data['brch\_light']+data['brch\_shoe']+data['brch\_other'])
135. fig, (axis1) = plt.subplots(1,1,figsize=(6,6))
136. data\_status = data[["problems","borough"]].groupby(['borough'],as\_index=False).mean()
137. sns.barplot(x='borough', y='problems', data=data\_status,order=[0,1,2,3,4],ax=axis1)
138. tips = data
139. fig, (axis3,axis4) = plt.subplots(1,2,figsize=(10,5))
140. sns.boxplot(x = 'borough' , y = 'problems', data = tips ,palette="PRGn", ax = axis3)
141. sns.boxplot(x = 'borough' , y = 'health', data = tips ,palette="PRGn", ax = axis4)
142. #Пройдем все районы и посмотрим на количство проблем у каждого дерева
143. data.groupby('borough')['problems'].sum()
144. #У кажого вида выделяем общую сумму проблем и выбираем виды с наименьшим числом проблем
145. arr = []
146. arr = data.groupby('spc\_latin')['problems'].sum()
147. for ind in range(133):
148. if arr[ind] < 50:
149. print(ind, arr[ind])