Задача 5

Условие

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

Задача: Провести анализ датасета и сделать обработку данных. Также ответить на следующие вопросы:

1 Сколько в датасете объектов и признаков? Дать описание каждому признаку, если

оно есть.

2 Сколько категориальных признаков, какие?

3 Столбец с макимальным количеством уникальных значений категориального

признака?

4 Есть ли бинарные признаки?

5 Какие числовые признаки?

6 Есть ли пропуски?

7 Сколько объектов с пропусками?

8 Столбец с максимальным количеством пропусков?

9 Есть ли на ваш взгляд выбросы, аномальные значения?

10 Столбец с максимальным средним значением после нормировки признаков через

стандартное отклонение?

11 Столбец с целевым признаком?

12 Сколько объектов попадает в тренировочную выборку при использовании

train\_test\_split с параметрами test\_size = 0.3, random\_state = 42?

13 Между какими признаками наблюдается линейная зависимость (корреляция)?

14 Сколько признаков достаточно для объяснения 90% дисперсии после применения

метода PCA?

15 Какой признак вносит наибольший вклад в первую компоненту?

Решение:

1. Отвечаем на вопрос 1 - Сколько в датасете объектов и признаков? Дать описание каждому признаку, если оно есть.
   1. 1)tree\_id - уникальный идентификационный номер дерева (целое число)
   2. 2)block\_id - Идентификационный номер блок-фасада, на который отображено дерево (целое число)
   3. 3)created\_at - Дата нанесения деревьев на карту (дата)
   4. 4)tree\_dbh - Диаметр на высоте груди дерева (целое число)
   5. 5)stump\_diam - Диаметр пня (целое число)
   6. 6)curb\_loc - Расположено ли дерево вдоль бордюра или со смещением от него (Текст Значения домена: OnCurb OffsetFromCurb)
   7. 7)status - Статус дерева (Текст Доменные значения: Живой, Пень, Мертвый)
   8. 8)health - Здоровье деревьев (Текст Доменные ценности: Хорошо, Неплохо, Плохо)
   9. 9)spc\_latin - Научное / латинское название вида дерева (Текст)
   10. 10)spc\_common - Общее название породы деревьев (Текст)
   11. 11)steward - Количество замеченных признаков управления (Текст Значения доменов: 1 или 2 ,3 или 4, 4 или больше, Ничего)
   12. 12)guards - Наличие и тип охраны деревьев (Текст Доменные ценности: Вредные Полезно Нет Не уверен)
   13. 13)sidewalk - Повреждение тротуара в непосредственной близости от дерева (Текст Значения домена: Повреждение, Не повреждено)
   14. 14)user\_type - Категория пользователя, который собрал эту точку дерева (Текст Доменные ценности: Волонтер, TreesCount Персонал, Персонал NYC Parks)
   15. 15)root\_stone - Проблемы с корнями, вызванные брусчаткой в ложе дерева (Текст Значения домена: Да Нет)
   16. 16)root\_grate - Проблемы с корнями, вызванные металлическими решетками (Текст Значения домена: Да Нет)
   17. 17)root\_other - Наличие других проблем с корнями (Текст Значения домена: Да Нет)
   18. 18)trunk\_wire - Проблемы со стволом, вызванные тросом или проводами (Текст Значения домена: Да Нет)
   19. 19)trnk\_light - Проблемы со стволом, вызванные освещением (Текст Значения домена: Да Нет)
   20. 20)trnk\_other - Наличие других проблем со стволом (Текст Значения домена: Да Нет)
   21. 21)brch\_light - Проблемы с ветками, вызванные освещением или проводами (Текст Значения домена: Да Нет)
   22. 22)brch\_shoe - Проблемы с ветвями, вызванные обувью (Текст Значения домена: Да Нет)
   23. 23)brch\_other - Наличие других проблем с ветвями (Текст Значения домена: Да Нет)
   24. 24)address - Предполагаемый уличный адрес дерева (Текст)
   25. 25)zipcode - Почтовый индекс (целое число)
   26. 26)zip\_city - Город, полученный из почтового индекса (Текст)
   27. 27)cb\_num - Общественный совет (Целое число)
   28. 28)borocode - Геокодирование на основе местоположения точки дерева (Целое число Доменные значения: 1 (Манхэттен) 2 (Бронкс) 3 (Бруклин) 4 (Квинс) 5 (Статен-Айленд))
   29. 29)boroname - Геокодирование на основе местоположения точки дерева (Текст Доменные ценности: Манхэттен Бронкс, Бруклин Квинс, Статен-Айленд)
   30. 30)cncldist - Округ совета Нью-Йорка (Целое число)
   31. 31)st\_assem - Округ Ассамблеи штата Нью-Йорк (Целое число)
   32. 32)st\_senate - Сенатский округ штата Нью-Йорк (Целое число)
   33. 33)nta - Код района табуляции соседей (Текст)
   34. 34)nta\_name - Название зоны табуляции соседей (Текст)
   35. 35)boro\_ct - Переписной участок (Текст)
   36. 36)state - Государство (Текст)
   37. 37)latitude - Широта точки дерева (Двойной)
   38. 38)longitude - Долгота точки дерева (Двойной)
   39. 39)x\_sp - X координата точки дерева (Двойной)
   40. 40)y\_sp - Y координата точки дерева (Двойной)
2. ВОПРОС 2. Сколько категориальных признаков, какие?

Всего 9 категориальных признаков:

1. status Статус дерева
2. health Здоровье деревьев
3. steward Количество замеченных признаков управления
4. guards Наличие и тип охраны деревьев
5. user\_type Категория пользователя, который собрал эту точку дерева
6. borocode Геокодирование на основе местоположения точки дерева
7. boroname Геокодирование на основе местоположения точки дерева
8. ВОПРОС 3. Столбец с макимальным количеством уникальных значений категориального признака?

Максимальное число категориальных признаков у столбца borocode и borough

1. ВОПРОС 4. Есть ли бинарные признаки?

Да, есть:

1. curb\_loc
2. sidewalk
3. root\_stone
4. root\_grate
5. root\_other
6. trunk\_wire
7. trnk\_light
8. trnk\_other
9. brch\_light
10. brch\_shoe
11. brch\_other
12. ВОПРОС 5. Какие числовые признаки?
    1. tree\_id
    2. block\_id
    3. created\_at
    4. tree\_dbh
    5. stump\_diam
    6. zipcode
    7. cb\_num
    8. cncldist
    9. st\_assem
    10. st\_senate
    11. latitude
    12. longitude
    13. x\_sp
    14. y\_sp
13. Исключим столбцы с уникальными значениями и включающие результат других столбцов: tree\_id, created\_at, address, problems.
14. ВОПРОС 6. Есть ли пропуски?
    1. Воспользуемся командой info() , чтобы увидеть общую информацию о датасете.

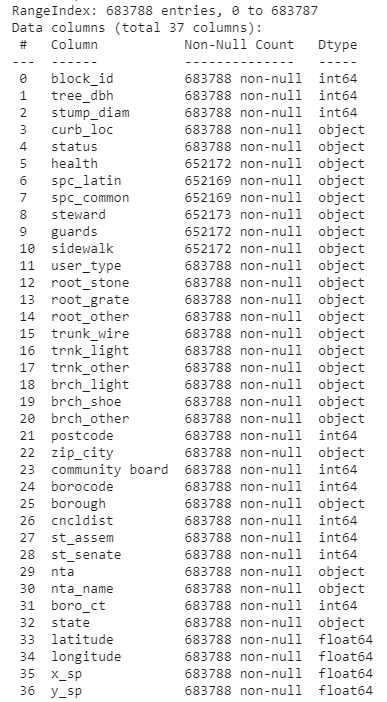


Рисунок 1. Информация о датасете.

Как можем наблюдать из Рисунка 1 – у нас есть пропуски.

1. ВОПРОС 7. Сколько объектов с пропусками?

Основываясь на результате info() - видим, что у 6-ти объектов есть пропуски.

1. ВОПРОС 8. Столбец с максимальным количеством пропусков?

Воспользуемся командой isnull(), для подсчета максимального числа пропусков.

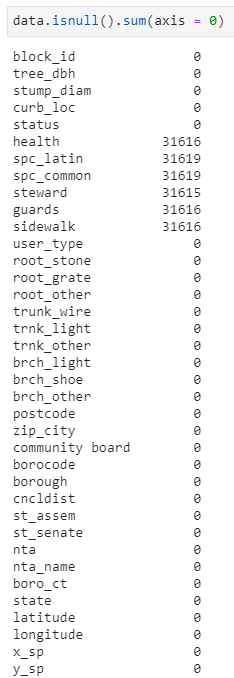


Рисунок 2. Результат работы isnull()

Следовательно, столбец/столбцы с наибольшим/им числом/ами пропусков - это spc\_latin = 31619 и spc\_common = 31619

1. ВОПРОС 9. Есть ли на ваш взгляд выбросы, аномальные значения?

Аномальных значений и выбросов не замечено. Смотри приложение 1.

1. ВОПРОС 10. Столбец с максимальным средним значением после нормировки признаков через стандартное отклонение? 

Рисунок 3. Поиск столбца с максимальным средним значением после нормировки признаков, через стандартное отклонение.

1. ВОПРОС 11. Столбец с целевым признаком?

Так как, мне надо выделить районы, в которых состояние деревьев аномально хорошее или плохое; выделить наилучший район по состоянию деревьев; выделить виды деревьев (или более общие элементы классификации), обладающие наилучшими показателями «здоровья», то будем оценивать столбцы:

1. 1)status - Указывает, является ли дерево живым, стоящим мертвым или пнем.
2. zip\_city - Город, полученный из почтового индекса
3. 3)root\_stone - Проблемы с корнями, вызванные брусчаткой в ложе дерева
4. 4)root\_grate - Проблемы с корнями, вызванные металлическими решетками
5. 5)root\_other - Наличие других проблем с корнями
6. 6)trunk\_wire - Проблемы со стволом, вызванные тросом или проводами
7. 7)trnk\_light - Проблемы со стволом, вызванные освещением
8. 8)trnk\_other - Наличие других проблем со стволом
9. 9)brch\_light - Проблемы с ветками, вызванные освещением или проводами
10. 10)brch\_shoe - Проблемы с ветвями, вызванные обувью
11. 11)brch\_other - Наличие других проблем с ветвями
12. 12)spc\_common - Общее название породы деревьев
13. Обрабатываем категориальные и бинарные признаки датасета. Смотри приложение 1.
14. ВОПРОС 12. Сколько объектов попадает в тренировочную выборку при использовании train\_test\_split с параметрами test\_size = 0.3, random\_state = 42?



Рисунок 4. Исключаем столбце с целевыми признаками из датасета для тренировки.

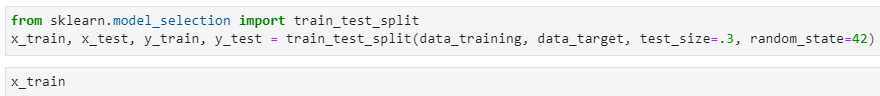


Рисунок 5. Разделение данных на тренировочные и тестовые.

В тренировочную выборку попадет 478651 объектов.

1. ВОПРОС 13. Между какими признаками наблюдается линейная зависимость (корреляция)?

Проверим коэффициент VIF (Variable Inflation Factors) для всех признаков. Если VIF > 10, то есть четкая линейная зависимость.

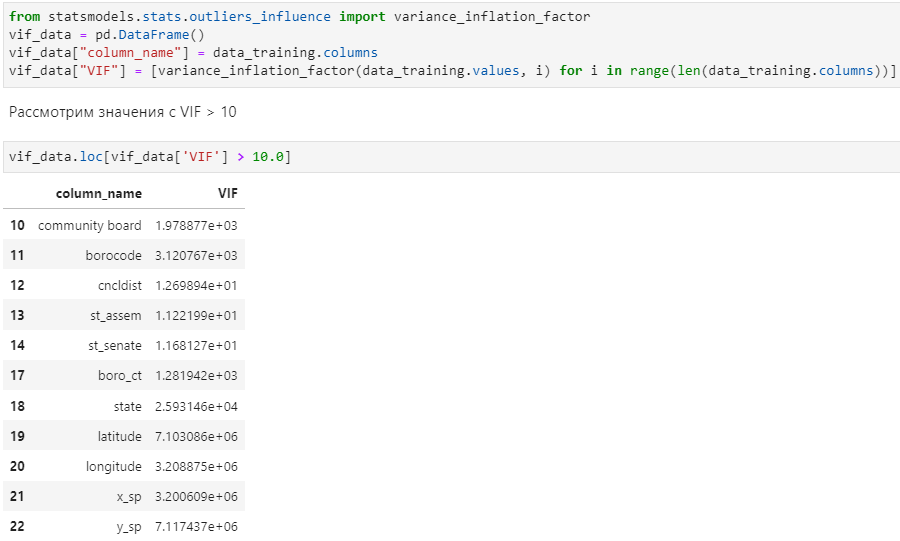


Рисунок 5. Поиск признаков с коэффициентов VIF больше 10.

1. ВОПРОС 14. Сколько признаков достаточно для объяснения 90% дисперсии после применения метода PCA?

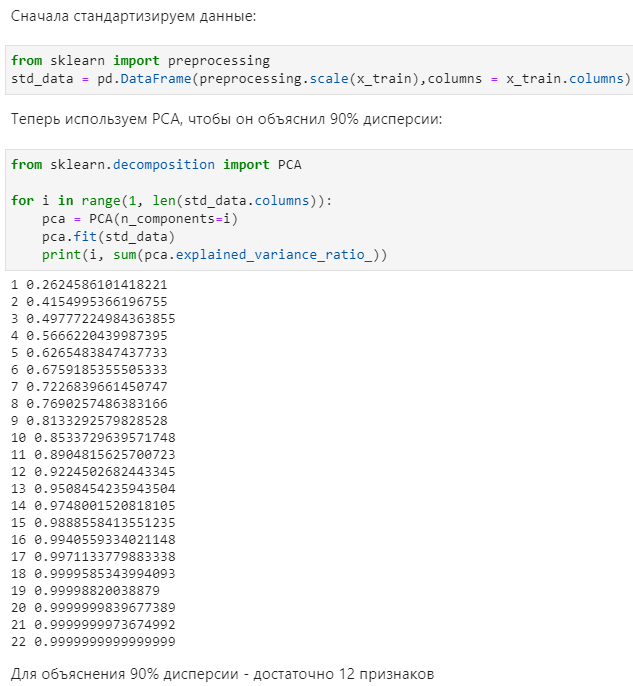


Рисунок 6. Применение метода главных компонентов.

1. Вопрос 15. Какой признак вносит наибольший вклад в первую компоненту?

Наибольший вклад в первую компоненту вносит 11-ый признак - borocode. Смотри приложение 1.

Приложение 1.

1. !pip install pandas
2. !pip install sklearn
3. import pandas as pd
4. import numpy as np
5. import warnings
6. warnings.filterwarnings('ignore')
7. data = pd.read\_csv('2015-street-tree-census-tree-data.csv')
8. #исключим столбец с уникальными значениями: tree\_id
9. data = data.drop('tree\_id', axis = 1)
10. #исключим столбец, включающий результаты других столбцов
11. data = data.drop('problems', axis = 1)
12. #исключим столбец с индивидуальными параметрами
13. data = data.drop('created\_at', axis = 1)
14. #исключим столбец с индивидуальными параметрами
15. data = data.drop('address', axis = 1)
16. #исключим столбцы с признаками, которые отсутствуют в документации
17. data = data.drop('council district', axis = 1)
18. data = data.drop('census tract', axis = 1)
19. data = data.drop('bin', axis = 1)
20. data = data.drop('bbl', axis = 1)
21. ctgrl\_priz = ['status','health','steward','guards','user\_type','borocode','borough']
22. for column in data.columns:
23. if column in ctgrl\_priz:
24. print(column, data[column].nunique())
25. data.info()
26. data.isnull().sum(axis = 0)
27. for column in data.columns:
28. print(data[column].value\_counts())
29. import math
30. arr = ['block\_id','tree\_dbh','stump\_diam','postcode','community board','cncldist','st\_assem','st\_senate','boro\_ct','latitude','longitude','x\_sp','y\_sp']
31. for column in arr:
32. data[column] = (data[column] - data[column].mean())/(math.sqrt(data[column].var()))
33. avrg = -math.inf
34. for column in arr:
35. if data[column].mean() > avrg:
36. avrg = data[column].mean()
37. print(avrg, column)
38. data['state'] = np.where(data['state'] == 'New York', 1, 0)
39. data.loc[data.spc\_latin == 'NaN', 'spc\_latin'] = 0
40. data.loc[data.spc\_common == 'NaN', 'spc\_common'] = 0
41. data.loc[data.steward == 'NaN', 'steward'] = 0
42. data.loc[data.guards == 'NaN', 'guards'] = 0
43. data.loc[data.sidewalk == 'NaN', 'sidewalk'] = 0
44. #Расположено ли дерево вдоль бордюра или со смещением от него
45. data['curb\_loc'] = np.where(data['curb\_loc'] == 'OnCurb', 1, 0)
46. #Повреждение тратуара в непосредственной близости от дерева
47. data['sidewalk'] = np.where(data['sidewalk'] == 'Damage', 1, 0)
48. #Указывает на наличие проблемы с корнями, вызванной брусчаткой в ложе дерева
49. data['root\_stone'] = np.where(data['root\_stone'] == 'Yes', 1, 0)
50. #Указывает на наличие проблемы с корнями, вызванной металлическими решетками в ложе дерева
51. data['root\_grate'] = np.where(data['root\_grate'] == 'Yes', 1, 0)
52. #Указывает на наличие других корневых проблем
53. data['root\_other'] = np.where(data['root\_other'] == 'Yes', 1, 0)
54. #Указывает на наличие проблемы со стволом, вызванной проволокой или веревкой, обмотанной вокруг ствола
55. data['trunk\_wire'] = np.where(data['trunk\_wire'] == 'Yes', 1, 0)
56. #Указывает на наличие проблем со стволом, вызванных установленным на дереве освещением
57. data['trnk\_light'] = np.where(data['trnk\_light'] == 'Yes', 1, 0)
58. #Указывает на наличие других проблем с магистралью
59. data['trnk\_other'] = np.where(data['trnk\_other'] == 'Yes', 1, 0)
60. #Указывает на наличие проблемы с ветвями, вызванной светильниками (обычно струнными) или проводами в ветвях
61. data['brch\_light'] = np.where(data['brch\_light'] == 'Yes', 1, 0)
62. #Указывает на наличие проблемы с ветками, вызванной кроссовками в ветках
63. data['brch\_shoe'] = np.where(data['brch\_shoe'] == 'Yes', 1, 0)
64. #Указывает на наличие других проблем с ветвями
65. data['brch\_other'] = np.where(data['brch\_other'] == 'Yes', 1, 0)
66. #Город, полученный из почтового индекса
67. my\_set = set(data.zip\_city)
68. i = 0
69. for item in my\_set:
70. data['zip\_city'] = data['zip\_city'].replace(item, i)
71. i = i + 1
72. #Это название NTA, соответствующее области табуляции района по переписи населения США 2010 года, в которую попадает точка дерева
73. my\_set = set(data.nta\_name)
74. i = 0
75. for item in my\_set:
76. data['nta\_name'] = data['nta\_name'].replace(item, i)
77. i = i + 1
78. #Это код NTA, соответствующий району, в который попадает точка дерева, согласно переписи населения США 2010 года.
79. my\_set = set(data.nta)
80. i = 0
81. for item in my\_set:
82. data['nta'] = data['nta'].replace(item, i)
83. i = i + 1
84. #Название района, в котором находится точка произрастания деревьев
85. my\_set = set(data.borough)
86. i = 0
87. for item in my\_set:
88. data['borough'] = data['borough'].replace(item, i)
89. i = i + 1
90. #Город, полученный из почтового индекса. Часто (но не всегда) это то же самое, что и район.
91. my\_set = set(data.postcode)
92. i = 0
93. for item in my\_set:
94. data['postcode'] = data['postcode'].replace(item, i)
95. i = i + 1
96. #Указывает, присутствует ли ограждение, и считает ли пользователь, что это полезное или вредное ограждение. Не регистрируется для мертвых деревьев и пней.
97. my\_set = set(data.guards)
98. i = 0
99. for item in my\_set:
100. data['guards'] = data['guards'].replace(item, i)
101. i = i + 1
102. #Это поле описывает категорию пользователя, который собрал данные этой точки дерева.
103. my\_set = set(data.user\_type)
104. i = 0
105. for item in my\_set:
106. data['user\_type'] = data['user\_type'].replace(item, i)
107. i = i + 1
109. #Указывает количество уникальных признаков бережного отношения, отмеченных для данного дерева. Не регистрируется для пней или мертвых деревьев.
110. my\_set = set(data.steward)
111. i = 0
112. for item in my\_set:
113. data['steward'] = data['steward'].replace(item, i)
114. i = i + 1
115. #Общее название вида, например, "красный клен"
116. my\_set = set(data.spc\_common)
117. i = 0
118. for item in my\_set:
119. data['spc\_common'] = data['spc\_common'].replace(item, i)
120. i = i + 1
122. #Научное название вида, например, "Acer rubrum".
123. my\_set = set(data.spc\_latin)
124. i = 0
125. for item in my\_set:
126. data['spc\_latin'] = data['spc\_latin'].replace(item, i)
127. i = i + 1
128. #Указывает, является ли дерево живым, стоящим мертвым или пнем.
129. data.loc[data.status == 'Alive', 'status'] = 2
130. data.loc[data.status == 'Stump', 'status'] = 1
131. data.loc[data.status == 'Dead', 'status'] = 0
132. #Здоровье деревьев
133. data.loc[data.health == 'Good', 'health'] = 2
134. data.loc[data.health == 'Fair', 'health'] = 1
135. data.loc[data.health == 'Poor', 'health'] = 0
136. data.loc[data.health == 'NaN', 'health'] = 0
137. data\_training = data.drop(['health',
138. 'borough',
139. 'root\_stone',
140. 'root\_grate',
141. 'root\_other',
142. 'trunk\_wire',
143. 'trnk\_light',
144. 'trnk\_other',
145. 'brch\_light',
146. 'brch\_shoe',
147. 'zip\_city',
148. 'brch\_other',
149. 'status',
150. 'spc\_common'],
151. axis = 1)
152. data\_target = data[['health',
153. 'borough',
154. 'root\_stone',
155. 'root\_grate',
156. 'root\_other',
157. 'trunk\_wire',
158. 'trnk\_light',
159. 'trnk\_other',
160. 'brch\_light',
161. 'brch\_shoe',
162. 'zip\_city',
163. 'brch\_other',
164. 'status',
165. 'spc\_common']]
166. data\_training.info()
167. from sklearn.model\_selection import train\_test\_split
168. x\_train, x\_test, y\_train, y\_test = train\_test\_split(data\_training, data\_target, test\_size=.3, random\_state=42)
169. x\_train
170. from statsmodels.stats.outliers\_influence import variance\_inflation\_factor
171. vif\_data = pd.DataFrame()
172. vif\_data["column\_name"] = data\_training.columns
173. vif\_data["VIF"] = [variance\_inflation\_factor(data\_training.values, i) for i in range(len(data\_training.columns))]
174. vif\_data.loc[vif\_data['VIF'] > 10.0]
175. from sklearn import preprocessing
176. std\_data = pd.DataFrame(preprocessing.scale(x\_train),columns = x\_train.columns)
177. from sklearn.decomposition import PCA
178. for i in range(1, len(std\_data.columns)):
179. pca = PCA(n\_components=i)
180. pca.fit(std\_data)
181. print(i, sum(pca.explained\_variance\_ratio\_))
182. pca = PCA(n\_components=12)
183. x = pca.fit(std\_data)
184. first\_component = pca.components\_[0]
185. i = 0
186. answ = first\_component[i]
187. for j in range(0, len(first\_component)):
188. if abs(first\_component[j]) > abs(answ):
189. answ = first\_component[j]
190. i = j
191. print(i, answ)
192. std\_data.iloc[:, 11]