**Московский государственный технический университет им. Н.Э.Баумана**

|  |  |
| --- | --- |
| Утверждаю: |  |
| Гапанюк Ю.Е. | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

**НИР по обработке и анализу данных**

пояснительная записка

(наименование документа)

9

(количество листов)

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | |
| студентка группы ИУ5-34М | | Абросимова Н.Г. |
|  | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_2021 г. | |

Москва - 2021

В качестве данных будет использоваться датасет, содержащий информацию об услугах, предоставляемых IT-предприятием. Будет проведён анализ данных и подбор рекомендаций по выбранной услуге.

**import** numpy **as** np

**import** pandas **as** pd

**import** seaborn **as** sns

**import** matplotlib.pyplot **as** plt

**from** typing **import** Dict, Tuple

**from** scipy **import** stats

**from** sklearn.feature\_extraction.text **import** CountVectorizer, TfidfVectorizer

**from** sklearn.neighbors **import** KNeighborsRegressor, KNeighborsClassifier

**from** sklearn.model\_selection **import** GridSearchCV, RandomizedSearchCV

**from** sklearn.metrics **import** confusion\_matrix

**from** sklearn.metrics.pairwise **import** cosine\_similarity, euclidean\_distances, manhattan\_distances

**from** surprise **import** SVD, Dataset, Reader

**from** matplotlib\_venn **import** venn2

**%matplotlib** inline

sns**.**set(style**=**"ticks")

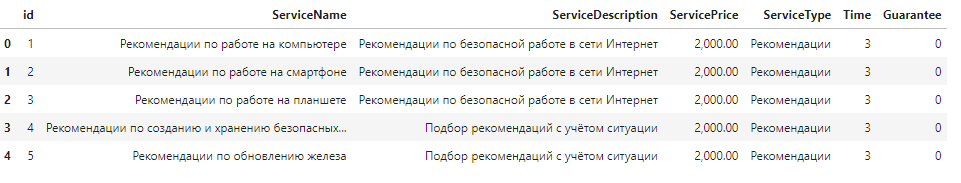
data**=**pd**.**read\_csv('ITService.csv', sep**=**",")

*#размер датасета*

data**.**shape

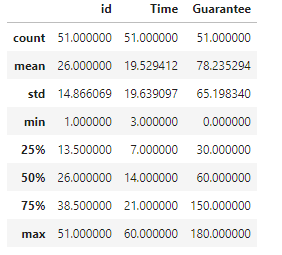
(51, 7)

data**.**head()



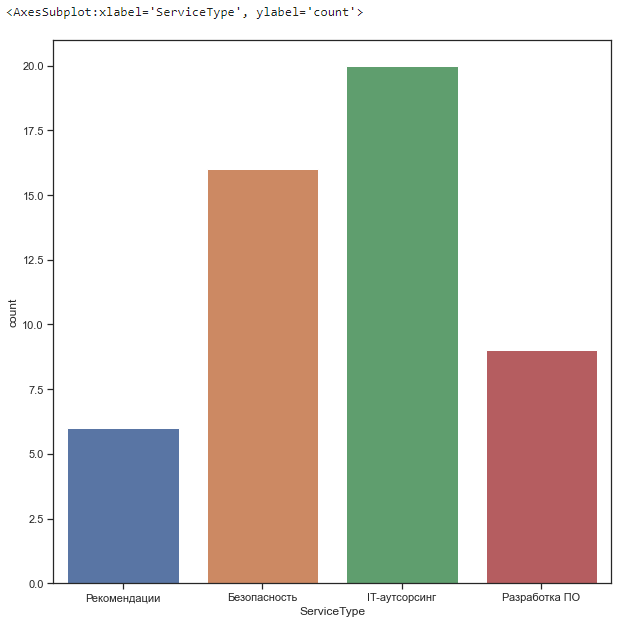
*# Основные статистические характеристки набора данных*

data**.**describe()



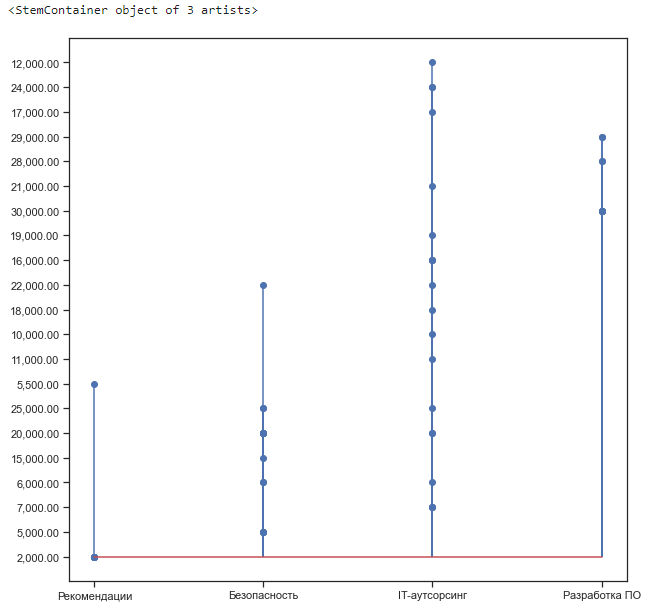
plt**.**figure(figsize**=**(10,10))

sns**.**countplot(x**=**"ServiceType", data**=**data)



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

plt**.**stem(data['ServiceType'], data['ServicePrice'])

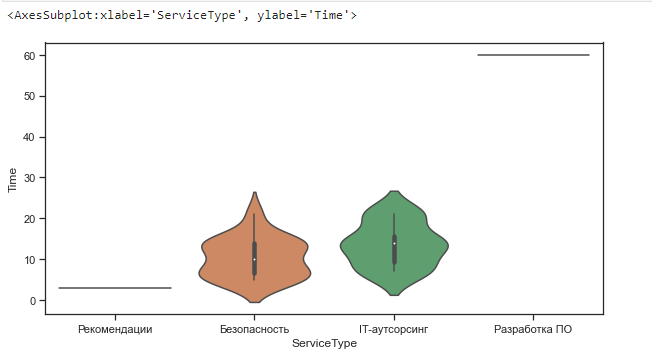


plt**.**figure(figsize**=**(10,5))

x**=**data['ServiceType']

y**=**data['Time']

sns**.**violinplot(x, y)



*# Колонки с пропусками*

data**.**isnull()**.**sum()

id 0

ServiceName 0

ServiceDescription 0

ServicePrice 0

ServiceType 0

Time 0

Guarantee 0

dtype: int64

Пропусков нет

data**.**dtypes

id int64

ServiceName object

ServiceDescription object

ServicePrice object

ServiceType object

Time int64

Guarantee int64

dtype: object

ServiceName**=** data['ServiceName']**.**values

ServiceName[0:5]

array(['Рекомендации по работе на компьютере',

'Рекомендации по работе на смартфоне',

'Рекомендации по работе на планшете',

'Рекомендации по созданию и хранению безопасных паролей',

'Рекомендации по обновлению железа'], dtype=object)

ServicePrice**=** data['ServicePrice']**.**values

ServicePrice[0:5]

array(['2,000.00', '2,000.00', '2,000.00', '2,000.00', '2,000.00'],

dtype=object)

ServiceType**=** data['ServiceType']**.**values

ServiceType[0:5]

array(['Рекомендации', 'Рекомендации', 'Рекомендации', 'Рекомендации',

'Рекомендации'], dtype=object)

Time**=** data['Time']**.**values

Time[0:5]

array([3, 3, 3, 3, 3], dtype=int64)

Guarantee**=** data['Guarantee']**.**values

Guarantee[0:5]

array([0, 0, 0, 0, 0], dtype=int64)

**%%time**

tfidfv **=** TfidfVectorizer()

matrix **=** tfidfv**.**fit\_transform(ServiceName)

matrix

Wall time: 8.27 ms

Out[18]:

<51x140 sparse matrix of type '<class 'numpy.float64'>'

with 230 stored elements in Compressed Sparse Row format>

Реализуем класс для формирования рекомендаций на основе метода ближайших соседей:

**class** SimpleKNNRecommender:

**def** \_\_init\_\_(self, X\_matrix, X\_ServiceType, X\_ServiceName, X\_ServicePrice, X\_Time, X\_Guarantee):

"""

Входные параметры:

X\_matrix - обучающая выборка (матрица объект-признак)

"""

*#Сохраняем параметры в переменных объекта*

self**.**\_X\_matrix **=** X\_matrix

self**.**df **=** pd**.**DataFrame(

{'ServiceType': pd**.**Series(X\_ServiceType, dtype**=**'str'),

'ServiceName': pd**.**Series(X\_ServiceName, dtype**=**'str'),

'ServicePrice': pd**.**Series(X\_ServicePrice, dtype**=**'str'),

'Time': pd**.**Series(X\_Time, dtype**=**'int64'),

'Guarantee': pd**.**Series(X\_Guarantee, dtype**=**'int64'),

'dist': pd**.**Series([], dtype**=**'float')})

**def** recommend\_for\_single\_object(self, K: int, \

X\_matrix\_object, cos\_flag **=** **True**, manh\_flag **=** **False**):

"""

Метод формирования рекомендаций для одного объекта.

Входные параметры:

K - количество рекомендуемых соседей

X\_matrix\_object - строка матрицы объект-признак, соответствующая объекту

cos\_flag - флаг вычисления косинусного расстояния

manh\_flag - флаг вычисления манхэттэнского расстояния

Возвращаемое значение: K найденных соседей

"""

scale **=** 1000000

*# Вычисляем косинусную близость*

**if** cos\_flag:

dist **=** cosine\_similarity(self**.**\_X\_matrix, X\_matrix\_object)

self**.**df['dist'] **=** dist **\*** scale

res **=** self**.**df**.**sort\_values(by**=**'dist', ascending**=False**)

*# Не учитываем рекомендации с единичным расстоянием,*

*# так как это искомый объект*

res **=** res[res['dist'] **<** scale]

**else**:

**if** manh\_flag:

dist **=** manhattan\_distances(self**.**\_X\_matrix, X\_matrix\_object)

**else**:

dist **=** euclidean\_distances(self**.**\_X\_matrix, X\_matrix\_object)

self**.**df['dist'] **=** dist **\*** scale

res **=** self**.**df**.**sort\_values(by**=**'dist', ascending**=True**)

*# Не учитываем рекомендации с единичным расстоянием,*

*# так как это искомый объект*

res **=** res[res['dist'] **>** 0.0]

*# Оставляем К первых рекомендаций*

res **=** res**.**head(K)

**return** res

ServiceName[5]

'Диагностика на наличие угроз и вредоносных программ'

mc\_matrix **=** matrix[5]

mc\_matrix

<1x140 sparse matrix of type '<class 'numpy.float64'>'

with 6 stored elements in Compressed Sparse Row format>

skr1 **=** SimpleKNNRecommender(matrix, ServiceType, ServiceName, ServicePrice, Time, Guarantee)

rec1 **=** skr1**.**recommend\_for\_single\_object(5, mc\_matrix)

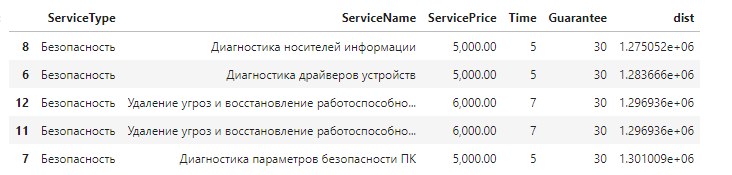
rec1



*# При поиске с помощью Евклидова расстояния*

rec2 **=** skr1**.**recommend\_for\_single\_object(5, mc\_matrix, cos\_flag **=** **False**)

rec2



*# Манхэттэнское расстояние*

rec3 **=** skr1**.**recommend\_for\_single\_object(5, mc\_matrix,

cos\_flag **=** **False**, manh\_flag **=** **True**)

rec3



Были построены графики, показывающие зависимость между типом услуг и ценой, типом услуг и временем выполнения, распространённость типов услуг. Была составлена система подбора рекомендаций на основе метода ближайших соседей, фильтрация проходила на основе содержания. Схожесть определялась на основе косинусного сходства, с помощью Евклидова расстояния, с помощью Манхэттэнского расстояния. Первые два метода показали наиболее точные результаты.