

Департамент образования и науки города Москвы  
Государственное автономное образовательное учреждение высшего  
образования города Москвы  
«Московский городской педагогический университет»  
Институт цифрового образования  
Департамент информатики, управления и технологий

**ДИСЦИПЛИНА:**

Инструменты для хранения и обработки больших данных

**Лабораторная работа 6.1**

«Обработка данных с использованием Apache Spark»

Выполнил(а): Морозова В.А. группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2024

## Цель и задачи работы:

1. Познакомиться с понятием «большие данные» и способами их обработки;
2. Познакомиться с инструментом Apache Spark и возможностями, которые он предоставляет для обработки больших данных.
3. Получить навыки выполнения разведочного анализа данных использованием pyspark.

Для начала импортируем необходимые библиотеки (рисунок 1)



```
✓ [1] # Import libraries
import pyspark
from pyspark import SparkConf
from pyspark.sql import SparkSession

from pyspark.sql.window import Window
from pyspark.sql.functions import udf, isnan, min, max, sum, count, desc, expr, avg
from pyspark.sql.types import IntegerType, LongType

from pyspark.ml.feature import StandardScaler, VectorAssembler, MinMaxScaler
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder, CrossValidatorModel
from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier, RandomForestClassifier, LogisticRegression, GBClassifier, LogisticRegressionModel, GBClassificationModel, RandomForestClassificationModel

import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import numpy as np
import pandas as pd

✓ [3] from google.colab import drive
drive.mount('content/drive')
Mounted at /content/drive

✓ [4] ! pip install pyspark
Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.3)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
```

Рисунок 1. Импорт необходимых библиотек

Далее подключаемся к спарку (рисунок 2)

```

[7] # Импорт модулей, связанных с PySpark.
import pyspark
from pyspark.rdd import RDD
from pyspark.sql import Row
from pyspark.sql import DataFrame
from pyspark.sql import SparkSession
from pyspark.sql import SQLContext
from pyspark.sql import functions
from pyspark.sql.functions import lit, desc, col, size, array_contains\
, isnan, udf, hour, array_min, array_max, countDistinct
from pyspark.sql.types import *

MAX_MEMORY = '15G'

conf = pyspark.SparkConf().setMaster("local[*]") \
    .set('spark.executor.heartbeatInterval', 10000) \
    .set('spark.network.timeout', 10000) \
    .set('spark.core.connection.ack.wait.timeout', "3600") \
    .set("spark.executor.memory", MAX_MEMORY) \
    .set("spark.driver.memory", MAX_MEMORY)

def init_spark():
    spark = SparkSession \
        .builder \
        .appName("Pyspark guide") \
        .config(conf=conf) \
        .getOrCreate()
    return spark

spark = init_spark()
filename_data = 'insurance.csv'
df = spark.read.csv(filename_data, mode="DROPMALFORMED", header = True)
print('Data frame type: ' + str(type(df)))

```

Data frame type: <class 'pyspark.sql.dataframe.DataFrame'>

Рисунок 2. Подключение к спарку

Затем идет обзор данных (рисунок 3).

```

[8] df.printSchema()
df.show()

```

```

root
 |-- age: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- bmi: string (nullable = true)
 |-- children: string (nullable = true)
 |-- smoker: string (nullable = true)
 |-- region: string (nullable = true)
 |-- charges: string (nullable = true)

```

age	sex	bmi	children	smoker	region	charges
19	female	27.9	0	yes	southwest	16884.924
18	male	33.77	1	no	southeast	1725.5523
28	male	33	3	no	southeast	4449.462
33	male	22.705	0	no	northwest	21984.47061
32	male	28.88	0	no	northwest	3866.8552
31	female	25.74	0	no	southeast	3756.6216
46	female	33.44	1	no	southeast	8240.5896
37	female	27.74	3	no	northwest	7281.5056
37	male	29.83	2	no	northeast	6406.4107
60	female	25.84	0	no	northwest	28923.13692
25	male	26.22	0	no	northeast	2721.3208
62	female	26.29	0	yes	southeast	27808.7251
23	male	34.4	0	no	southwest	1826.843
56	female	39.82	0	no	southeast	11090.7178
27	male	42.13	0	yes	southeast	39611.7577
19	male	24.6	1	no	southwest	1837.237
52	female	30.78	1	no	northeast	10797.3362
23	male	23.845	0	no	northeast	2395.17155
56	male	40.3	0	no	southwest	10602.385
30	male	35.3	0	yes	southwest	36837.467

only showing top 20 rows

Рисунок 3. Обзор данных

Потом надо узнать типы данных (рисунок 4)

```
df.printSchema()
pd.DataFrame(df.dtypes, columns = ['Column Name', 'Data type'])
```

```
root
|-- age: string (nullable = true)
|-- sex: string (nullable = true)
|-- bmi: string (nullable = true)
|-- children: string (nullable = true)
|-- smoker: string (nullable = true)
|-- region: string (nullable = true)
|-- charges: string (nullable = true)
```

	Column Name	Data type
0	age	string
1	sex	string
2	bmi	string
3	children	string
4	smoker	string
5	region	string
6	charges	string

Рисунок 4. Узнавание типов данных

Изменение типов данных и вывод количества строк показано на рисунке

5.

```
[11] pandas_df = df.withColumn("age", df["age"].cast(IntegerType())) \
      .withColumn("bmi", df["bmi"].cast(FloatType())) \
      .withColumn("children", df["children"].cast(IntegerType())) \
      .withColumn("charges", df["charges"].cast(FloatType()))
pandas_df.printSchema()
pd.DataFrame(pandas_df.dtypes, columns = ['Column Name', 'Data type'])
```

```
root
|-- age: integer (nullable = true)
|-- sex: string (nullable = true)
|-- bmi: float (nullable = true)
|-- children: integer (nullable = true)
|-- smoker: string (nullable = true)
|-- region: string (nullable = true)
|-- charges: float (nullable = true)
```

	Column Name	Data type
0	age	int
1	sex	string
2	bmi	float
3	children	int
4	smoker	string
5	region	string
6	charges	float

```
[12] pandas_df.describe().toPandas()
print(f'Общее количество {pandas_df.count()} строк, печатаем несколько первых строк:')
pandas_df.limit(5).toPandas()
```

Общее количество 1338 строк, печатаем несколько первых строк:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900000	0	yes	southwest	16884.923828
1	18	male	33.770000	1	no	southeast	1725.552246
2	28	male	33.000000	3	no	southeast	4449.461914
3	33	male	22.705000	0	no	northwest	21984.470703
4	32	male	28.879999	0	no	northwest	3866.855225

Рисунок 5. Изменение типов данных

Проверка наличия нулевых значений продемонстрировано на рисунке 6.  
Как можно заметить, их нет

Проверка наличия нулевых значений

```
[13] pd.DataFrame(pandas_df.dtypes, columns = ['Column Name','Data type'])
```

	Column Name	Data type
0	age	int
1	sex	string
2	bmi	float
3	children	int
4	smoker	string
5	region	string
6	charges	float

```
str_col = ['Sex', 'Smoker', 'Region']
num_col = ['Children', 'Age']
mis_val = {}
for index, column in enumerate(pandas_df.columns):
    if column in str_col:
        missing_count = pandas_df.filter(col(column).eqNullSafe(None) | col(column).isNull()).count()
        mis_val.update({column:missing_count})
    if column in num_col:
        mis_count = pandas_df.where(col(column).isin([0,None,np.nan])).count()
        mis_val.update({column:missing_count})

mis_df = pd.DataFrame.from_dict(mis_val)
mis_df
```

	0
0	

Рисунок 6. Проверка наличия нулевых значений

Расчет статистических показателей (квартили, дисперсия, средняя и медиана) на рисунке 7.

Статистические показатели

```
[30] from pyspark.sql.functions import col, avg, percentile_approx, variance
      quartiles = pandas_df.select(
          percentile_approx("Age", 0.25).alias("1_q"),
          percentile_approx("Age", 0.5).alias("2_q"),
          percentile_approx("Age", 0.75).alias("3_q")
      )
      quartiles.show()
```

	1_q	2_q	3_q
0	27	39	51

```
[31] stat = pandas_df.select(
      avg("Age").alias("Средний возраст"),
      variance("Age").alias("Дисперсия"),
      percentile_approx("Age", 0.5).alias("Медиана")
  )
  stat.show()
```

	Средний возраст	Дисперсия	Медиана
0	39.20702541106129	197.40138665754355	39

Рисунок 7. Расчет статистических показателей

Параметры по полу, региону и отношению к курению относятся к категориальным данным, что недопустимо для модели, их нужно преобразовать в 0 и 1 для регрессионного анализа и получения точных результатов

```
[ ] 1 df["Sex_num"] = df["sex"].map({'male':0, 'female':1})
    2 df["Smoker_num"] = df["smoker"].map({'yes':1, 'no':0})
    3 df["Region_num"] = df["region"].map({'northwest':1, 'southeast':2, 'southwest':3, 'northeast':4})
```

1 df

	age	sex	bmi	children	smoker	region	charges	Sex_num	Smoker_num	Region_num
0	19	female	27.900	0	yes	southwest	16884.92400	1	1	3
1	18	male	33.770	1	no	southeast	1725.55230	0	0	2
2	28	male	33.000	3	no	southeast	4449.46200	0	0	2
3	33	male	22.705	0	no	northwest	21984.47061	0	0	1
4	32	male	28.880	0	no	northwest	3866.85520	0	0	1
...	...	...	...	...	...	...	...	...	...	...

Рисунок 8. Преобразование данных для анализа

✓ Анализ по каждой характеристике отдельно простой линейной регрессии

```
1 model_age = sm.ols(formula="charges ~ age", data=df).fit()
2 model_age.summary()
```

OLS Regression Results

Dep. Variable:	charges	R-squared:	0.089
Model:	OLS	Adj. R-squared:	0.089
Method:	Least Squares	F-statistic:	131.2
Date:	Tue, 09 Apr 2024	Prob (F-statistic):	4.89e-29
Time:	11:31:54	Log-Likelihood:	-14415.
No. Observations:	1338	AIC:	2.883e+04
Df Residuals:	1336	BIC:	2.884e+04
Df Model:	1		
Covariance Type:	nonrobust		



	coef	std err	t	P> t	[0.025	0.975]
Intercept	3165.8850	937.149	3.378	0.001	1327.440	5004.330
age	257.7226	22.502	11.453	0.000	213.579	301.866

Omnibus: 399.600 Durbin-Watson: 2.033  
 Prob(Omnibus): 0.000 Jarque-Bera (JB): 864.239  
 Skew: 1.733 Prob(JB): 2.15e-188  
 Kurtosis: 4.869 Cond. No. 124.

Рисунок 9. Анализ зависимости затрат от возраста

Результат означает, что каждый дополнительный год увеличения возраста увеличивает расходы на 257.7 денежных единиц

$R^2=8,9\%$  это значит, что модель объясняет 8,9% изменчивости расходов. Это очень маленькое значение, указывающее на нехорошее соответствие модели данным.


Прогноз расходов на медицину\_Морозова Валерия
☆


Файл
Изменить
Вид
Вставка
Среда выполнения
Инструменты
Справка

Код
+ Текст

▶

```

1 model_bmi = sm.ols(formula="charges ~ bmi", data=df).fit()
2 model_bmi.summary()

```

↔

OLS Regression Results									
<b>Dep. Variable:</b>	charges	<b>R-squared:</b>	0.039						
<b>Model:</b>	OLS	<b>Adj. R-squared:</b>	0.039						
<b>Method:</b>	Least Squares	<b>F-statistic:</b>	54.71						
<b>Date:</b>	Tue, 09 Apr 2024	<b>Prob (F-statistic):</b>	2.46e-13						
<b>Time:</b>	11:31:59	<b>Log-Likelihood:</b>	-14451.						
<b>No. Observations:</b>	1338	<b>AIC:</b>	2.891e+04						
<b>Df Residuals:</b>	1336	<b>BIC:</b>	2.892e+04						
<b>Df Model:</b>	1								
<b>Covariance Type:</b> nonrobust									
	coef	std err	t	P> t	[0.025	0.975]			
<b>Intercept</b>	1192.9372	1664.802	0.717	0.474	-2072.974	4458.849			
<b>bmi</b>	393.8730	53.251	7.397	0.000	289.409	498.337			
<b>Omnibus:</b>	261.030	<b>Durbin-Watson:</b>	1.983						
<b>Prob(Omnibus):</b>	0.000	<b>Jarque-Bera (JB):</b>	431.091						
<b>Skew:</b>	1.297	<b>Prob(JB):</b>	2.45e-94						
<b>Kurtosis:</b>	4.004	<b>Cond. No.</b>	160.						

Рисунок 10. Зависимость затрат от ИМТ

Показывает, что увеличение ИМТ на каждую единицу увеличивает расходы на 393.8 денежных единиц.

$R^2=3,9\%$  это значит, что модель объясняет 3,9% изменчивости расходов. Это очень маленькое значение, указывающее на нехорошее соответствие модели данным.

```
[ ] 1 model_smoker = sm.ols(formula="charges ~ Smoker_num", data=df).fit()
    2 model1_smoker.summary()
```



OLS Regression Results

Dep. Variable:	charges	R-squared:	0.620
Model:	OLS	Adj. R-squared:	0.619
Method:	Least Squares	F-statistic:	2178.
Date:	Tue, 09 Apr 2024	Prob (F-statistic):	8.27e-283
Time:	11:32:10	Log-Likelihood:	-13831.
No. Observations:	1338	AIC:	2.767e+04
Df Residuals:	1336	BIC:	2.768e+04
Df Model:	1		

Covariance Type: nonrobust

	coef	std err	t	P> t	[0.025	0.975]
Intercept	8434.2683	229.014	36.829	0.000	7985.002	8883.535
Smoker_num	2.362e+04	506.075	46.665	0.000	2.26e+04	2.46e+04

Omnibus: 135.996 Durbin-Watson: 2.025  
 Prob(Omnibus): 0.000 Jarque-Bera (JB): 212.201  
 Skew: 0.727 Prob(JB): 8.34e-47  
 Kurtosis: 4.300 Cond. No. 2.60

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Итог значительно увеличивает расходы для курильщиков по сравнению с некурящими, при прочих равных условиях.

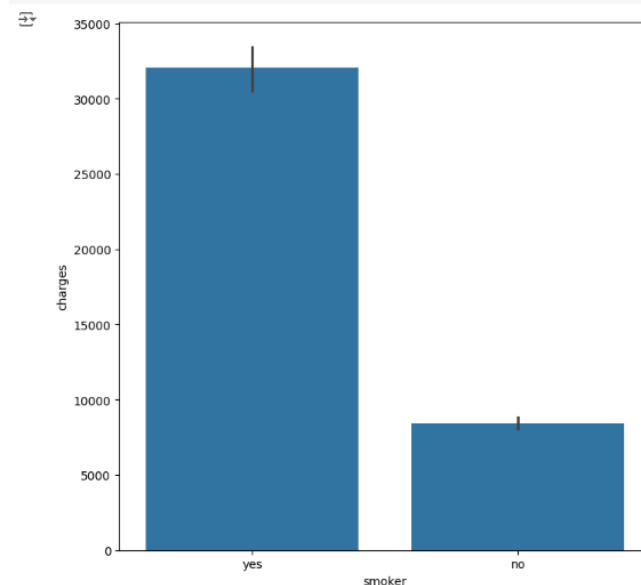
## Рисунок 11. Зависимость затрат от статуса курения

Итог значительно увеличивает расходы для курильщиков по сравнению с некурящими, при прочих равных условиях.

## Визуализация

Графики

```
1 plt.figure(figsize=(8,8))
2 sns.barplot(data = df, x="smoker", y="charges")
3 plt.show()
```



## Рисунок 12. Количество курящих и не курящих



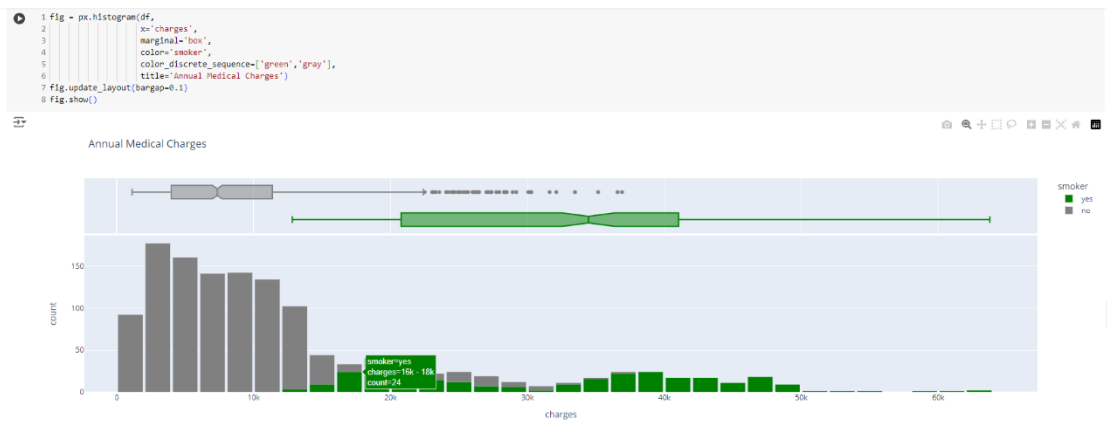


Рисунок 13. Демонстрация зависимости количества и статуса к курению и затрат на лечение

Количество людей по возрасту на рисунке 14. Как можно заметить, больше всего людей в возрасте до 20 лет

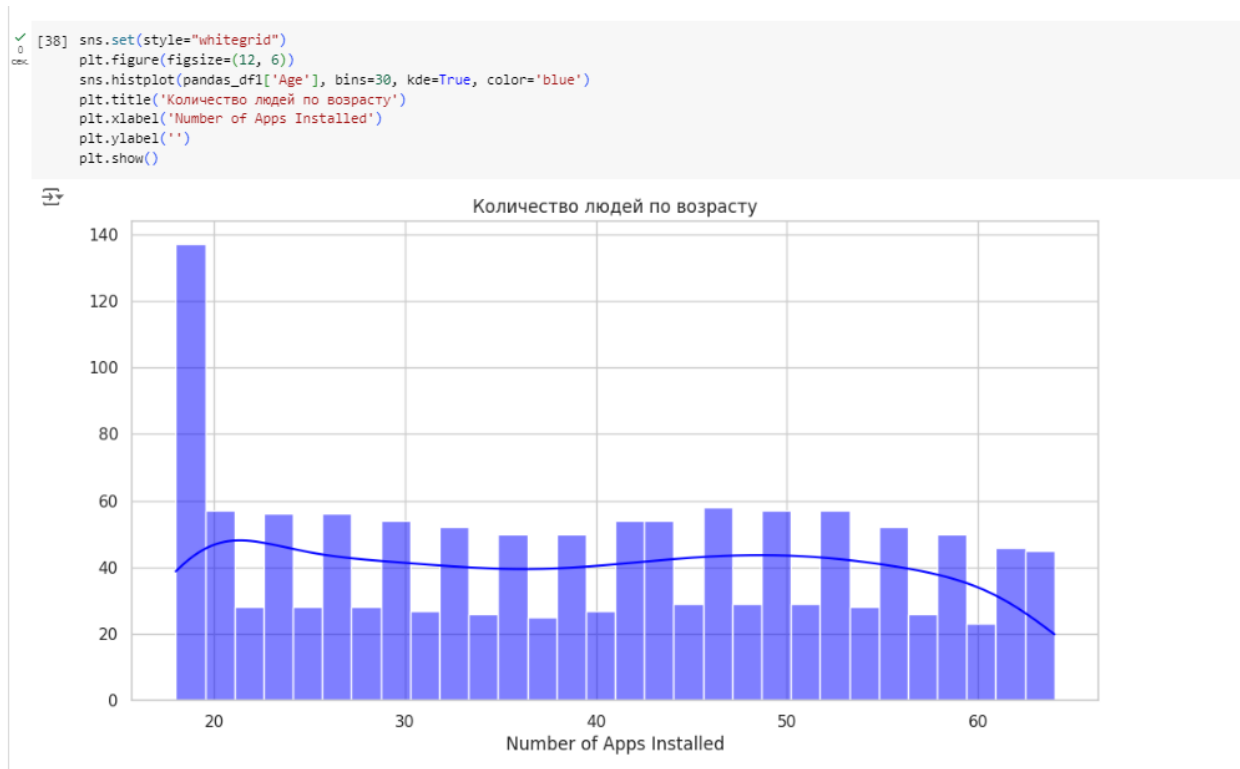


Рисунок 14. Количество людей по возрасту

Затем идет UDF: первый на возвращение отношения charges к age, а второй на классификацию, а второй на классификацию индекса масса тела (рисунок 15).

```
[39] def calculate_metric(charges, age):  
    if age == 0:  
        return 0.0  
    return charges / age # например, мы возвращаем отношение charges к age  
  
# Определение UDF для классификации BMI  
def classify_bmi(bmi):  
    if bmi < 18.5:  
        return "Underweight"  
    elif 18.5 <= bmi < 24.9:  
        return "Normal"  
    elif 25 <= bmi < 29.9:  
        return "Overweight"  
    else:  
        return "Obese"
```

Рисунок 15. UDF

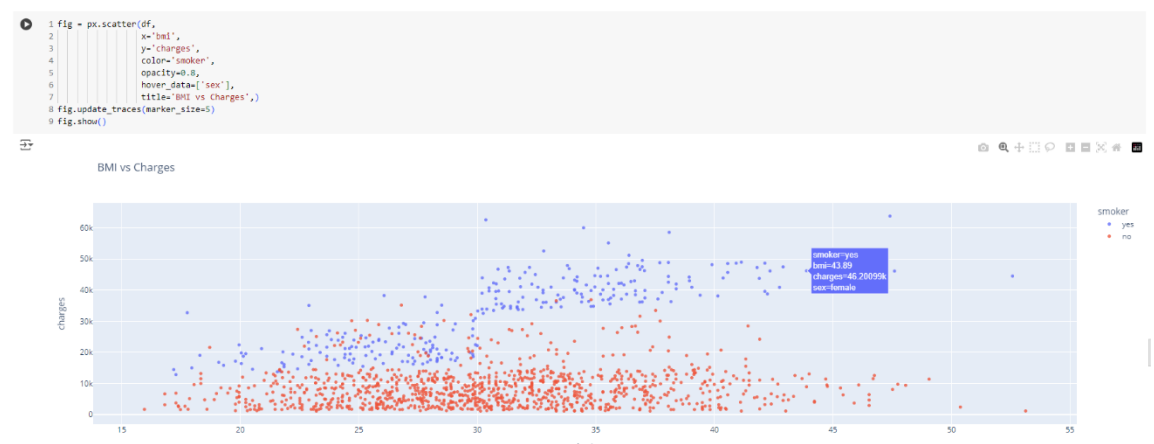


Рисунок 16. Отображение взаимозависимости ИМТ, курения и затрат

## Вывод:

Чтобы выяснить, есть ли зависимость между демографическими показателями человека и расходами денежных средств, которые он готов потратить на медицинское обслуживание, был проанализирован набор данных, включающий такие метрики, связанные с клиентом, как пол, возраст, индекс массы тела, количество детей, регион и статус в отношении курения.

1. При анализе типов признаков в датасете было установлено наличие как количественных, так и категориальных признаков.
2. Пропущенных значений в представленном датасете не оказалось.
3. Были рассчитаны основные статистические показатели, такие как средние значения, медианы, квартильные значения и дисперсия.
4. Проведен анализ линейной регрессии, выявлена взаимосвязь.
5. Было реализовано несколько пользовательских функций (UDF).
6. Проведена наглядная визуализация.

Apache Spark — это целостная вычислительная система с набором библиотек для параллельной обработки данных на кластерах компьютеров.

Spark производит вычисления в реальном времени и обеспечивает низкую задержку благодаря их резидентному выполнению. В отличие от Hadoop, более ориентирован для обработки данных с использованием оперативной памяти на основе распределения данных.