Отчёт по проекту

“Регрессионный анализ возрастных изменений моллюсков на “Abalone Age Prediction” наборе данных”

1. Введение

В кухнях всего мира морские ушки (Abalone) считаются деликатесом, а из их раковин делают украшения из-за их переливающегося блеска. Размер морских улиток варьируется от маленьких до очень крупных. Из-за экономической ценности и спроса на морское ушко его часто собирают на фермах, поэтому для прогнозирования его возраста необходимы физические измерения. При традиционном методе раковину прорезают по конусу, окрашивают, а затем пересчитывают под микроскопом их кольца, чтобы определить ее возраст. Для определения возраста морского ушка используется трудоемкий процесс прорезания раковины через конус, ее окрашивания и подсчета колец под микроскопом.

2. Постановка задачи

Данных проект предназначен для решения проблемы необходимости трудоёмкого процесса традиционного измерения возраста моллюсков Abalone.

Задача состоит в поиске оптимальной модели для решения задачи при помощи инструментов Apache PySpark и различных вспомогательных библиотек Python.

Был использован регрессионный анализ для определения возраста моллюсков по их размерам, полу и весу без использования данных полученных при прорезании раковин, что поможет облегчить процесс определения возраста моллюсков.

3. Описание набора данных

Набор данных Abalone Age Prediction представляет собой таблицу со следующими атрибутами:

- длина (наибольшее измерение ракушки) в мм (15-163)
- диаметр (перпендикуляр длине) в мм (11-130)
- высота в мм (4-50)
- общий вес в граммах (0.4-570)
- вес очищенного (граммы) (0.2-297)
- вес ракушки (граммы) (0.3-201)

- вес внутренности (граммы) (0.1-152)
- пол (мужской, женский, младенец)
- количество колец/возраст - (1-29 лет)

На следующем скриншоте можно видеть общую схему набора данных:

# length Length	# diameter Diameter	# height Height	# whole-weight Whole weight	# shucked-weight Shucked weight	# viscera-weight Viscera weight	# shell-weight Shell weight	# sex_F Sex - onehot encoded
15 163	11 130	0 226	0.4 565	0.2 298	0.1 152	0.3 201	0 1
124	97	41	243.8	77.5	50.1	77.0	0
109	88	27	183.7	85.8	40.3	47.5	1
136	108	38	324.6	143.3	70.8	94.3	0
110	88	33	172.1	62.4	33.8	60.0	0
78	60	20	53.3	22.1	11.8	16.8	0
123	99	31	257.3	87.0	58.6	64.9	0
106	82	28	136.2	61.9	28.3	36.7	0
105	83	34	166.5	55.1	33.7	62.0	1
113	88	25	160.4	71.9	36.5	43.0	0
76	55	19	48.5	21.2	9.7	42.0	0
137	109	36	353.6	149.9	78.4	97.0	1
70	54	15	43.0	20.0	7.2	13.0	0
120	101	38	225.8	87.7	51.2	72.0	1
117	94	29	191.3	80.5	47.3	53.0	0
138	112	43	343.8	136.0	59.8	94.0	1
110	85	28	190.4	97.9	38.9	43.7	1
125	98	35	246.6	111.3	49.4	73.0	1
121	91	29	195.5	93.6	35.5	55.0	1
116	90	47	214.2	60.0	41.2	79.0	1
127	99	36	319.2	123.4	63.4	74.0	1
114	87	34	174.6	76.4	36.6	54.1	0
119	91	30	208.8	103.6	44.1	54.0	0
145	113	42	428.5	206.0	97.4	100.6	1

3. Ход работы

Гипотеза:

Возможно создать модель, достаточно точную для замены традиционного метода измерения возраста моллюсков Abalone при использовании регрессионных моделей Apache PySpark.

Для решения задачи были выбраны следующие модели:

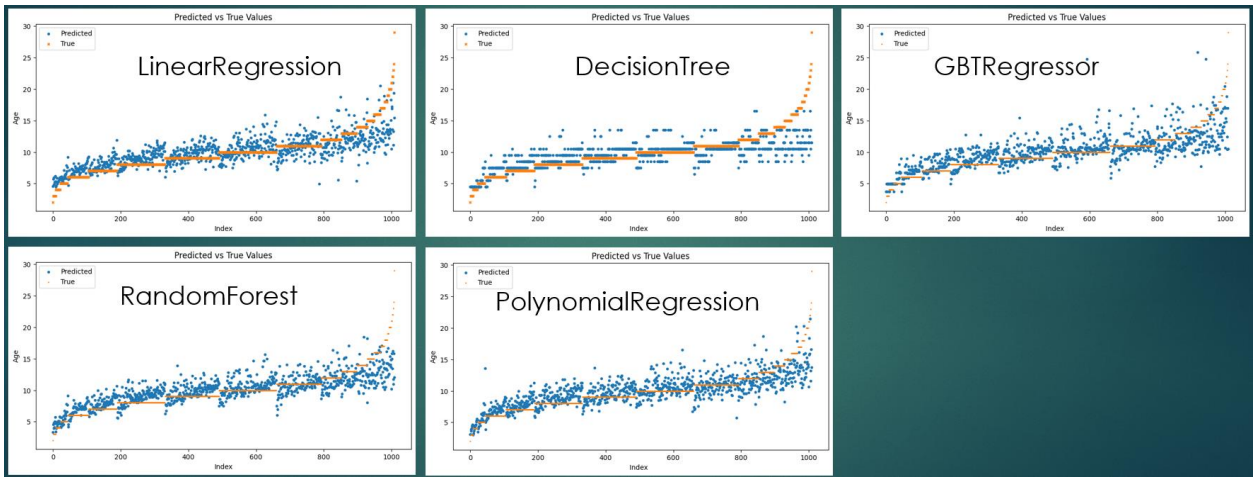
- Линейная регрессия
- Решающие деревья
- Случайный лес
- Полиномиальная регрессия
- Деревья на повышении градиента

Каждый метод может оказаться подходящим для данного датасета из-за своей структуры, поэтому был проведён анализ, подобраны параметры и было проведено сравнение результатов. Для более гибких моделей был использован GridSearch для поиска оптимальных гиперпараметров.

Для измерения ошибки была использована мера различия RMSE, в итоге были получены следующие результаты:

- Линейная регрессия (RMSE = 2.4818)
- Решающие деревья (RMSE = 2.4818)
- Случайный лес (RMSE = 2.10413 с GridSearch и ~2.25 без использования GridSearch)
- Полиномиальная регрессия (RMSE = 2.0575)
- Деревья на повышении градиента (Gradient-Boosted Trees) (RMSE = 2.3222)

Визуализация истинных и предсказанных значений при помощи Matplotlib на следующем скриншоте:



Оранжевым показаны истинные значения, а голубые точки – предсказанное значение возраста.

На этом скриншоте можно видеть визуализацию полученных результатов, где можно видеть хоть и небольшое, но превосходство модели PolynomialRegression.

Фрагмент кода с предобработкой датасета и наиболее точной моделью на следующий скриншотах:

Установка pyspark, импорт библиотек, создание структур и подготовка данных

```
[ ] #pip install pyspark
```

```
[ ] from pyspark.sql import SparkSession
    from pyspark.sql.types import StructType, StructField, DoubleType, IntegerType, BooleanType
    from pyspark.sql.functions import col, rand, when
    from pyspark.ml.feature import VectorAssembler, StandardScaler, PolynomialExpansion
    from pyspark.ml import Pipeline
    from pyspark.ml.regression import LinearRegression, RandomForestRegressor
    from pyspark.ml.evaluation import RegressionEvaluator, MulticlassClassificationEvaluator
    from pyspark.ml.tuning import ParamGridBuilder, CrossValidator, TrainValidationSplit
    from pyspark.ml.regression import GBTRegressor


    #sc.setLogLevel("OFF")

    spark = SparkSession.builder.appName("AbaloneAgePrediction").getOrCreate()

    #Структура для инициализации
    initSchema = StructType([
        StructField("Length", IntegerType(), True),
        StructField("Diameter", IntegerType(), True),
        StructField("Height", IntegerType(), True),
        StructField("WholeWeight", DoubleType(), True),
        StructField("ShuckedWeight", DoubleType(), True),
        StructField("VisceraWeight", DoubleType(), True),
        StructField("ShellWeight", DoubleType(), True),
        StructField("Sex_F", IntegerType(), True),
        StructField("Sex_I", IntegerType(), True),
        StructField("Sex_M", IntegerType(), True),
        StructField("Age", DoubleType(), True)
    ])

    #
    # goalSchema = StructType([
    #     StructField("Length", IntegerType(), True),
    #     StructField("Diameter", IntegerType(), True),
    #     StructField("Height", IntegerType(), True),
    #     StructField("WholeWeight", DoubleType(), True),
    #     StructField("ShuckedWeight", DoubleType(), True),
    #     StructField("VisceraWeight", DoubleType(), True),
    #     StructField("ShellWeight", DoubleType(), True),
    #     StructField("Age", DoubleType(), True),
    #     StructField("Gen", IntegerType(), True)
    # ])

    df = spark.read.csv("train_dataset.csv", header=True, schema=initSchema)
```

```
root
 |-- Length: integer (nullable = true)
 |-- Diameter: integer (nullable = true)
 |-- Height: integer (nullable = true)
 |-- WholeWeight: double (nullable = true)
 |-- ShuckedWeight: double (nullable = true)
 |-- VisceraWeight: double (nullable = true)
 |-- ShellWeight: double (nullable = true)
 |-- Sex_F: integer (nullable = true)
 |-- Sex_I: integer (nullable = true)
 |-- Sex_M: integer (nullable = true)
 |-- Age: double (nullable = true)

+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+
|Length|Diameter|Height|WholeWeight|ShuckedWeight|VisceraWeight|ShellWeight|Sex_F|Sex_I|Sex_M| Age|
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+
|   124|      97|    41|      243.8|         77.5|         50.1|       77.0|    0|    0|    1|14.0|
|   109|      88|    27|      183.7|         85.8|         40.3|       47.5|    1|    0|    0|10.0|
|   136|     108|    38|      324.6|        143.3|         70.8|       94.3|    0|    0|    1|12.0|
|   110|      88|    33|      172.1|         62.4|         33.8|       60.0|    0|    1|    0|17.0|
|    78|      60|    20|       53.3|         22.1|         11.8|       16.8|    0|    1|    0| 7.0|
|   123|      99|    31|      257.3|         87.0|         58.6|       64.9|    0|    0|    1|11.0|
|   106|      82|    28|      136.2|         61.9|         28.3|       36.7|    0|    0|    1| 6.0|
|   105|      83|    34|      166.5|         55.1|         33.7|       62.0|    1|    0|    0|13.0|
|   113|      88|    25|      160.4|         71.9|         36.5|       43.0|    0|    0|    1| 9.0|
|    76|      55|    19|       48.5|         21.2|          9.7|       42.0|    0|    1|    0| 6.0|
|   137|     109|    36|      353.6|        149.9|         78.4|       97.0|    1|    0|    0|16.0|
|    70|      54|    15|       43.0|         20.0|          7.2|       13.0|    0|    1|    0| 6.0|
|   120|     101|    38|      225.8|         87.7|         51.2|       72.0|    1|    0|    0|13.0|
|   117|      94|    29|      191.3|         80.5|         47.3|       53.0|    0|    0|    1| 9.0|
|   138|     112|    43|      343.8|        136.0|         59.8|       94.0|    1|    0|    0|17.0|
|   110|      85|    28|      190.4|         97.9|         38.9|       43.7|    1|    0|    0| 7.0|
|   125|      98|    35|      246.6|        111.3|         49.4|       73.0|    1|    0|    0|11.0|
|   121|      91|    29|      195.5|         93.6|         35.5|       55.0|    1|    0|    0| 9.0|
|   116|      90|    47|      214.2|         60.0|         41.2|       79.0|    1|    0|    0|14.0|
|   127|      99|    36|      319.2|        123.4|         63.4|       74.0|    1|    0|    0|11.0|
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+
only showing top 20 rows
```

Scaler

```
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withMean=True, withStd=True)
scalerModel = scaler.fit(output)
print(scalerModel)
scaledData = scalerModel.transform(output)
scaledData = scaledData.drop("features")
newDF = scaledData.withColumnRenamed("scaledFeatures", "features")
newDF.show(truncate=False)
```

```
StandardScalerModel: uid=StandardScaler_788c58a16d38, numFeatures=10, withMean=true, withStd=true
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
|Length|Diameter|Height|WholeWeight|ShuckedWeight|VisceraWeight|ShellWeight|Sex_F|Sex_I|Sex_M|Age |features
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
|124   |97      |41    |243.8      |77.5         |50.1         |77.0       |0    |0    |1    |14.0|[0.8015103931876265,0.7796883210240684,1.5429056271
|109   |88      |27    |183.7      |85.8         |40.3         |47.5       |1    |0    |0    |10.0|[0.18021616299603932,0.3284422046085462,-0.10292219
|136   |108     |38    |324.6      |143.3        |70.8         |94.3       |0    |0    |1    |12.0|[1.2985457773408962,1.3312113521985955,1.1902282376
|110   |88      |33    |172.1      |62.4         |33.8         |60.0       |0    |1    |0    |17.0|[0.22163577834214515,0.3284422046085462,0.602432588
|78    |60      |20    |53.3       |22.1         |11.8         |16.8       |0    |1    |0    |7.0 |[-1.1037919127332407,-1.075434602017523,-0.92583609
|123   |99      |31    |257.3      |87.0         |58.6         |64.9       |0    |0    |1    |11.0|[0.7600907778415207,0.8799652357830733,0.3673143289
|106   |82      |28    |136.2      |61.9         |28.3         |36.7       |0    |0    |1    |6.0 |[0.055957316957721895,0.02761146033153136,0.0146369
|105   |83      |34    |166.5      |55.1         |33.7         |62.0       |1    |0    |0    |13.0|[0.014537701611616084,0.07774991771103383,0.7199917
|113   |88      |25    |160.4      |71.9         |36.5         |43.0       |0    |0    |1    |9.0 |[0.3458946243804626,0.3284422046085462,-0.338040449
|76    |55      |19    |48.5       |21.2         |9.7          |42.0       |0    |1    |0    |6.0 |[-1.1866311434254524,-1.3261268889150353,-1.0433952
|137   |109     |36    |353.6      |149.9        |78.4         |97.0       |1    |0    |0    |16.0|[1.339965392687002,1.381349809578098,0.955109978065
|70    |54      |15    |43.0       |20.0         |7.2          |13.0       |0    |1    |0    |6.0 |[-1.4351488355020874,-1.3762653462945378,-1.5136317
|120   |101     |38    |225.8      |87.7         |51.2         |72.0       |1    |0    |0    |13.0|[0.6358319318032033,0.9802421505420783,1.1902282376
|117   |94      |29    |191.3      |80.5         |47.3         |53.0       |0    |0    |1    |9.0 |[0.5115730857648858,0.629272948885561,0.13219606935
|138   |112     |43    |343.8      |136.0        |59.8         |94.0       |1    |0    |0    |17.0|[1.3813850080331078,1.5317651817166054,1.7780238867
|110   |85      |28    |190.4      |97.9         |38.9         |43.7       |1    |0    |0    |7.0 |[0.22163577834214515,0.17802683247003875,0.01463693
|125   |98      |35    |246.6      |111.3        |49.4         |73.0       |1    |0    |0    |11.0|[0.8429300085337323,0.8298267784035709,0.8375508482
|121   |91      |29    |195.5      |93.6         |35.5         |55.0       |1    |0    |0    |9.0 |[0.677251547149309,0.4788575767470536,0.13219606935
|116   |90      |47    |214.2      |60.0         |41.2         |79.0       |1    |0    |0    |14.0|[0.47015347041878003,0.4287191193675511,2.248260406
|127   |99      |36    |319.2      |123.4        |63.4         |74.0       |1    |0    |0    |11.0|[0.9257692392259439,0.8799652357830733,0.9551099780
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
only showing top 20 rows
```


Scaler

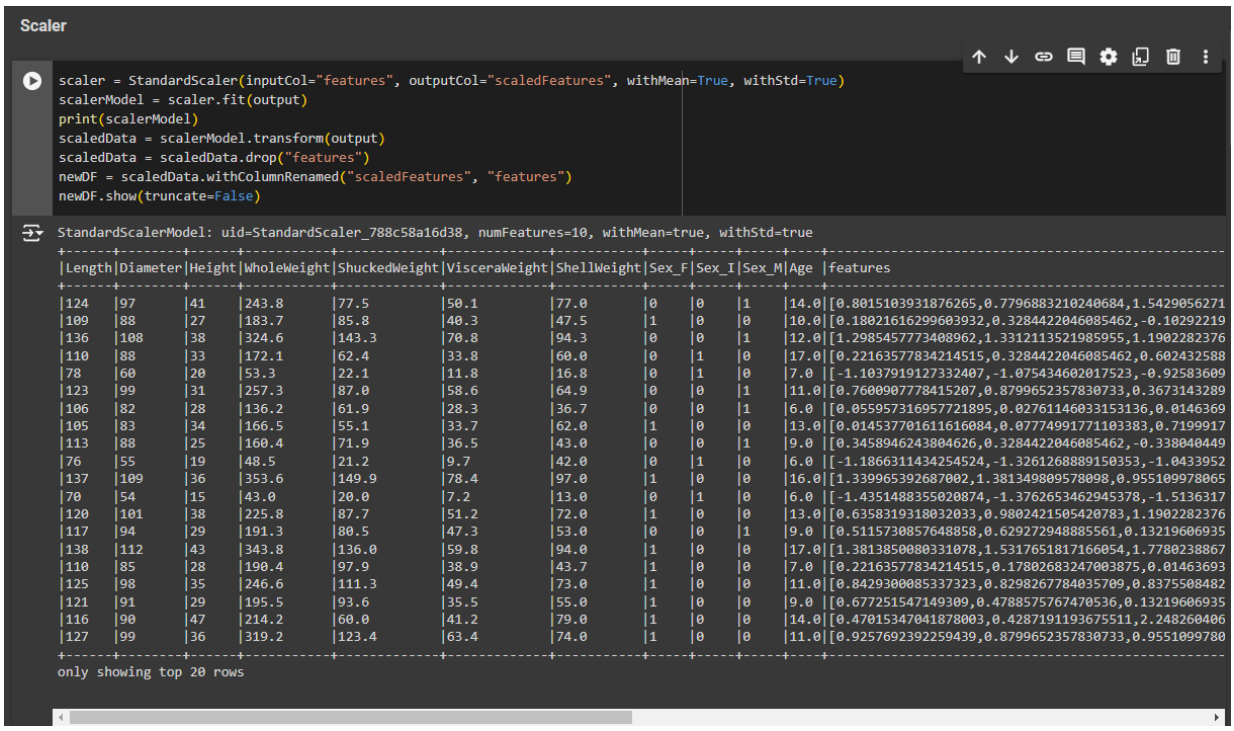
```
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withMean=True, withStd=True)
scalerModel = scaler.fit(output)
print(scalerModel)
scaledData = scalerModel.transform(output)
scaledData = scaledData.drop("features")
newDF = scaledData.withColumnRenamed("scaledFeatures", "features")
newDF.show(truncate=False)
```

```
StandardScalerModel: uid=StandardScaler_788c58a16d38, numFeatures=10, withMean=true, withStd=true
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
|Length|Diameter|Height|WholeWeight|ShuckedWeight|VisceraWeight|ShellWeight|Sex_F|Sex_I|Sex_M|Age |features
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
|124   |97      |41    |243.8      |77.5         |50.1         |77.0       |0    |0    |1    |14.0|[0.8015103931876265,0.7796883210240684,1.5429056271
|109   |88      |27    |183.7      |85.8         |40.3         |47.5       |1    |0    |0    |10.0|[0.18021616299603932,0.3284422046085462,-0.10292219
|136   |108     |38    |324.6      |143.3        |70.8         |94.3       |0    |0    |1    |12.0|[1.2985457773408962,1.3312113521985955,1.1902282376
|110   |88      |33    |172.1      |62.4         |33.8         |60.0       |0    |1    |0    |17.0|[0.22163577834214515,0.3284422046085462,0.602432588
|78    |60      |20    |53.3       |22.1         |11.8         |16.8       |0    |1    |0    |7.0 |[-1.1037919127332407,-1.075434602017523,-0.92583609
|123   |99      |31    |257.3      |87.0         |58.6         |64.9       |0    |0    |1    |11.0|[0.7600907778415207,0.8799652357830733,0.3673143289
|106   |82      |28    |136.2      |61.9         |28.3         |36.7       |0    |0    |1    |6.0 |[0.055957316957721895,0.02761146033153136,0.0146369
|105   |83      |34    |166.5      |55.1         |33.7         |62.0       |1    |0    |0    |13.0|[0.014537701611616084,0.07774991771103383,0.7199917
|113   |88      |25    |160.4      |71.9         |36.5         |43.0       |0    |0    |1    |9.0 |[0.3458946243804626,0.3284422046085462,-0.338040449
|76    |55      |19    |48.5       |21.2         |9.7          |42.0       |0    |1    |0    |6.0 |[-1.1866311434254524,-1.3261268889150353,-1.0433952
|137   |109     |36    |353.6      |149.9        |78.4         |97.0       |1    |0    |0    |16.0|[1.339965392687002,1.381349809578098,0.955109978065
|70    |54      |15    |43.0       |20.0         |7.2          |13.0       |0    |1    |0    |6.0 |[-1.4351488355020874,-1.3762653462945378,-1.5136317
|120   |101     |38    |225.8      |87.7         |51.2         |72.0       |1    |0    |0    |13.0|[0.6358319318032033,0.9802421505420783,1.1902282376
|117   |94      |29    |191.3      |80.5         |47.3         |53.0       |0    |0    |1    |9.0 |[0.5115730857648858,0.629272948885561,0.13219606935
|138   |112     |43    |343.8      |136.0        |59.8         |94.0       |1    |0    |0    |17.0|[1.3813850080331078,1.5317651817166054,1.7780238867
|110   |85      |28    |190.4      |97.9         |38.9         |43.7       |1    |0    |0    |7.0 |[0.22163577834214515,0.17802683247003875,0.01463693
|125   |98      |35    |246.6      |111.3        |49.4         |73.0       |1    |0    |0    |11.0|[0.8429300085337323,0.8298267784035709,0.8375508482
|121   |91      |29    |195.5      |93.6         |35.5         |55.0       |1    |0    |0    |9.0 |[0.677251547149309,0.4788575767470536,0.13219606935
|116   |90      |47    |214.2      |60.0         |41.2         |79.0       |1    |0    |0    |14.0|[0.47015347041878003,0.4287191193675511,2.248260406
|127   |99      |36    |319.2      |123.4        |63.4         |74.0       |1    |0    |0    |11.0|[0.9257692392259439,0.8799652357830733,0.9551099780
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
only showing top 20 rows
```


Scaler

```
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withMean=True, withStd=True)
scalerModel = scaler.fit(output)
print(scalerModel)
scaledData = scalerModel.transform(output)
scaledData = scaledData.drop("features")
newDF = scaledData.withColumnRenamed("scaledFeatures", "features")
newDF.show(truncate=False)
```

```
StandardScalerModel: uid=StandardScaler_788c58a16d38, numFeatures=10, withMean=true, withStd=true
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
|Length|Diameter|Height|WholeWeight|ShuckedWeight|VisceraWeight|ShellWeight|Sex_F|Sex_I|Sex_M|Age |features
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
|124   |97      |41    |243.8      |77.5         |50.1         |77.0       |0    |0    |1    |14.0|[0.8015103931876265,0.7796883210240684,1.5429056271
|109   |88      |27    |183.7      |85.8         |40.3         |47.5       |1    |0    |0    |10.0|[0.18021616299603932,0.3284422046085462,-0.10292219
|136   |108     |38    |324.6      |143.3        |70.8         |94.3       |0    |0    |1    |12.0|[1.2985457773408962,1.3312113521985955,1.1902282376
|110   |88      |33    |172.1      |62.4         |33.8         |60.0       |0    |1    |0    |17.0|[0.22163577834214515,0.3284422046085462,0.602432588
|78    |60      |20    |53.3       |22.1         |11.8         |16.8       |0    |1    |0    |7.0 |[-1.1037919127332407,-1.075434602017523,-0.92583609
|123   |99      |31    |257.3      |87.0         |58.6         |64.9       |0    |0    |1    |11.0|[0.7600907778415207,0.8799652357830733,0.3673143289
|106   |82      |28    |136.2      |61.9         |28.3         |36.7       |0    |0    |1    |6.0 |[0.055957316957721895,0.02761146033153136,0.0146369
|105   |83      |34    |166.5      |55.1         |33.7         |62.0       |1    |0    |0    |13.0|[0.014537701611616084,0.07774991771103383,0.7199917
|113   |88      |25    |160.4      |71.9         |36.5         |43.0       |0    |0    |1    |9.0 |[0.3458946243804626,0.3284422046085462,-0.338040449
|76    |55      |19    |48.5       |21.2         |9.7          |42.0       |0    |1    |0    |6.0 |[-1.1866311434254524,-1.3261268889150353,-1.0433952
|137   |109     |36    |353.6      |149.9        |78.4         |97.0       |1    |0    |0    |16.0|[1.339965392687002,1.381349809578098,0.955109978065
|70    |54      |15    |43.0       |20.0         |7.2          |13.0       |0    |1    |0    |6.0 |[-1.4351488355020874,-1.3762653462945378,-1.5136317
|120   |101     |38    |225.8      |87.7         |51.2         |72.0       |1    |0    |0    |13.0|[0.6358319318032033,0.9802421505420783,1.1902282376
|117   |94      |29    |191.3      |80.5         |47.3         |53.0       |0    |0    |1    |9.0 |[0.5115730857648858,0.629272948885561,0.13219606935
|138   |112     |43    |343.8      |136.0        |59.8         |94.0       |1    |0    |0    |17.0|[1.3813850080331078,1.5317651817166054,1.7780238867
|110   |85      |28    |190.4      |97.9         |38.9         |43.7       |1    |0    |0    |7.0 |[0.22163577834214515,0.17802683247003875,0.01463693
|125   |98      |35    |246.6      |111.3        |49.4         |73.0       |1    |0    |0    |11.0|[0.8429300085337323,0.8298267784035709,0.8375508482
|121   |91      |29    |195.5      |93.6         |35.5         |55.0       |1    |0    |0    |9.0 |[0.677251547149309,0.4788575767470536,0.13219606935
|116   |90      |47    |214.2      |60.0         |41.2         |79.0       |1    |0    |0    |14.0|[0.47015347041878003,0.4287191193675511,2.248260406
|127   |99      |36    |319.2      |123.4        |63.4         |74.0       |1    |0    |0    |11.0|[0.9257692392259439,0.8799652357830733,0.9551099780
+------+--------+------+-----------+-------------+-------------+-----------+-----+-----+-----+----+----------------------------------------------------------
only showing top 20 rows
```



Полный код можно найти в GitHub репозитории: [GitHub репозитории](#).

Также был проведён анализ различных решений других пользователей на сайте, где был найден набор данных Kaggle.com. В основном они использовали библиотеки pytorch или scykitlearn, использовались различные модели регрессии, KNN, кластеризация, случайный лес. Наиболее точным из них оказалось “Easy Regression to Determine Abalone Age”: [решение “Easy Regression to Determine Abalone Age”](#). Рейтинг RMSE = 2.1298 при использовании модели линейной регрессии и случайного леса.

4. Вывод

- Была найдена модель, позволяющая оценивать возраст моллюсков Abalone с точностью предсказания RMSE=~2.05, что позволяет использовать модель как замену традиционному методу измерения при отсутствии требования очень точного результата.
- Данная модель оказалась на ~0.06 RMSE точнее, чем аналогичные модели от других пользователей