Random-Forest

Начало работы:

1) Делаем форк
2) Клонируем репо на ПК
3) Открываем папку и переходим в каталог своего варианта

```
PS C:\Users\Roman\Desktop\bml-2024-tasks> cd ".\tasks\Random-Forest\Bank-Churn"
PS C:\Users\Roman\Desktop\bml-2024-tasks\tasks\Random-Forest\Bank-Churn>
```

4) Добавляем реализацию
5) Обучаем модель
6) Сохраняем в кэш для дальнейшего использования

```
from joblib import dump, load

# Сохранение модели
dump(model, 'model.joblib')
```

7) Тестируем обученную модель

Когда мы обучили модель её можно сохранить для дальнейшего использования

```
# Если модель уже обучена и сохранена
# Загрузка модели
model = load('model.joblib')
from goto import goto, label
goto .start
```

```
label .start
```

То есть воспользуемся обученной моделью из «кэша»

Описание алгоритма:

Random Forest — это широко используемый ансамблевый алгоритм машинного обучения, который повышает точность предсказаний, комбинируя результаты нескольких деревьев решений. Он особенно эффективен как для задач классификации, так и для задач регрессии благодаря своей способности снижать переобучение и улучшать стабильность модели.

Как работает Random Forest

Ансамбль деревьев решений: Random Forest строит «лес» из деревьев решений, где каждое дерево обучается на случайной подвыборке обучающих данных с использованием метода, известного как бустраппинг. Это означает, что данные отбираются с возвращением, что позволяет некоторым наблюдениям повторяться, а другим — отсутствовать.

Случайность признаков: При создании каждого дерева Random Forest вводит дополнительную случайность, выбирая случайную подвыборку признаков для разделения узлов. Этот подход гарантирует, что деревья будут некоррелированными и помогут захватить различные паттерны в данных, что приводит к улучшению обобщающей способности.

Агрегация предсказаний:

Для задач классификации окончательное предсказание делается путем голосования большинства среди всех деревьев.

Для задач регрессии предсказания усредняются для получения окончательного результата.

Описание результатов:

Исследовательский анализ данных (EDA)

```
PS C:\Users\Roman\Desktop\bml-2024-tasks\tasks\Random-Forest\Bank-Churn> python random-forest.py
```

Вывод нескольких строчек содержимого файла data.csv

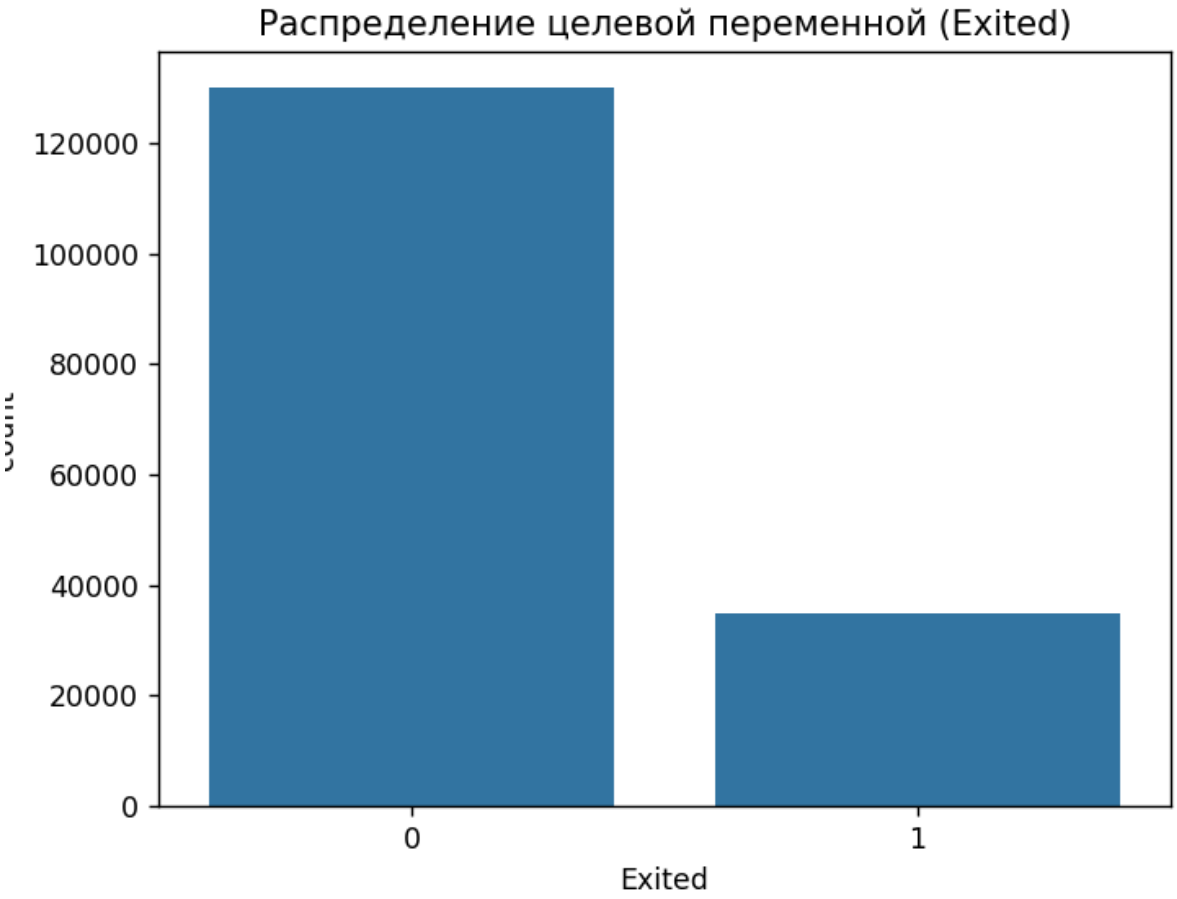
	id	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	0	15674932	Okwudilichukwu	668	France	Male	33.0	3	0.00	2	1.0	0.0	181449.97	0
1	1	15749177	Okwudiliolisa	627	France	Male	33.0	1	0.00	2	1.0	1.0	49503.50	0
2	2	15694510	Hsueh	678	France	Male	40.0	10	0.00	2	1.0	0.0	184866.69	0
3	3	15741417	Kao	581	France	Male	34.0	2	148882.54	1	1.0	1.0	84560.88	0
4	4	15766172	Chiemenam	716	Spain	Male	33.0	5	0.00	2	1.0	1.0	15068.83	0

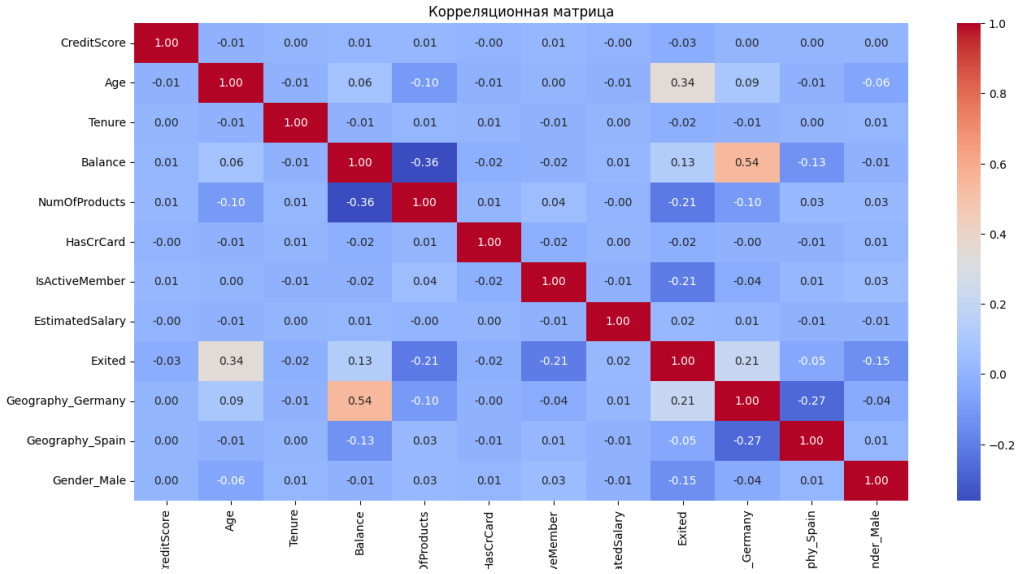
Количество «вхождений» в файл и непустых значений столбцов

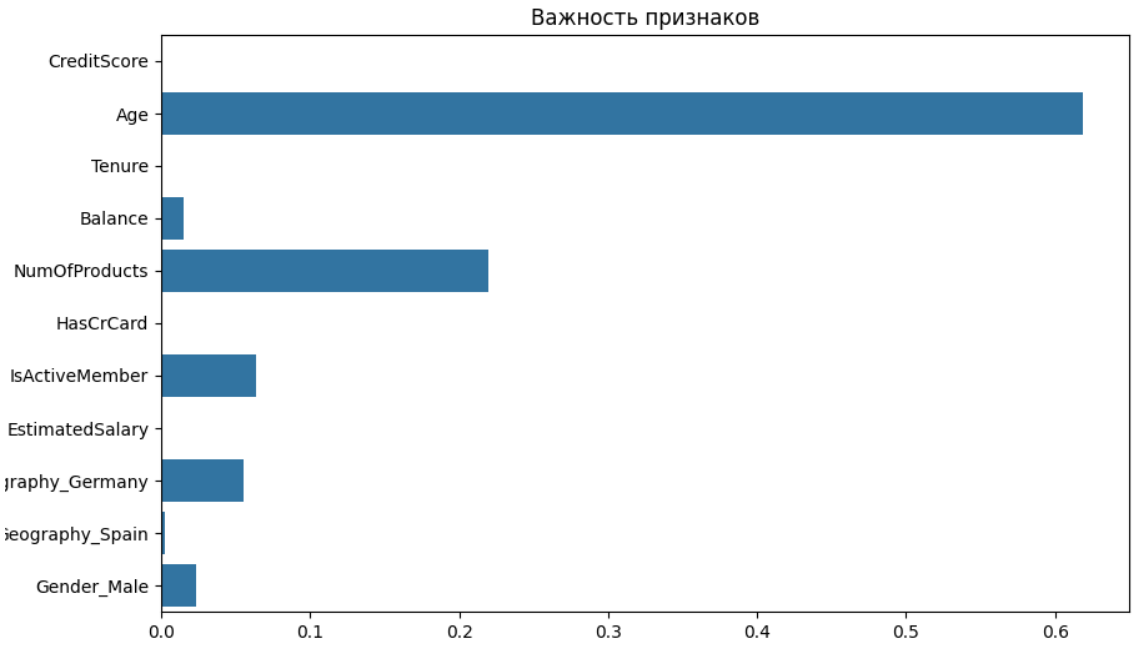
```
RangeIndex: 165034 entries, 0 to 165033
Data columns (total 14 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   id               165034 non-null  int64
 1   CustomerId       165034 non-null  int64
 2   Surname          165034 non-null  object
 3   CreditScore      165034 non-null  int64
 4   Geography        165034 non-null  object
 5   Gender           165034 non-null  object
 6   Age              165034 non-null  float64
 7   Tenure           165034 non-null  int64
 8   Balance          165034 non-null  float64
 9   NumOfProducts    165034 non-null  int64
 10  HasCrCard        165034 non-null  float64
 11  IsActiveMember   165034 non-null  float64
 12  EstimatedSalary  165034 non-null  float64
 13  Exited           165034 non-null  int64
dtypes: float64(5), int64(6), object(3)
memory usage: 17.6+ MB
```

Описание статистических характеристик

	id	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	165034.0000	1.650340e+05	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000	165034.000000
mean	82516.5000	1.569201e+07	656.454373	38.125888	5.020353	55478.086689	1.554455	0.753954	0.497770	112574.822734	0.211599
std	47641.3565	7.139782e+04	80.103340	8.867205	2.806159	62817.663278	0.547154	0.430707	0.499997	50292.865585	0.408443
min	0.0000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.000000	0.000000	11.580000	0.000000
25%	41258.2500	1.563314e+07	597.000000	32.000000	3.000000	0.000000	1.000000	1.000000	0.000000	74637.570000	0.000000
50%	82516.5000	1.569017e+07	659.000000	37.000000	5.000000	0.000000	2.000000	1.000000	0.000000	117948.000000	0.000000
75%	123774.7500	1.575682e+07	710.000000	42.000000	7.000000	119939.517500	2.000000	1.000000	1.000000	155152.467500	0.000000
max	165033.0000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.000000	1.000000	199992.480000	1.000000







Результаты теста:

```
Матрица ошибок:
[[25882   170]
 [ 5929  1026]]

Отчет о классификации:
              precision    recall  f1-score   support

           0       0.81      0.99      0.89     26052
           1       0.86      0.15      0.25      6955

    accuracy                           0.82     33007
   macro avg       0.84      0.57      0.57     33007
weighted avg       0.82      0.82      0.76     33007
```

Правильно предсказано что клиент перестанет пользоваться 25882

Неправильно предсказано что клиент продолжит пользоваться 166

Неправильно предсказано что клиент перестанет пользоваться 5837

Правильно предсказано что клиент продолжит пользоваться 1118