

## I. Логика сопоставления столбцов:

- 1) Разберемся с 3 датами. Логично, что самая «ранняя» из 3 дат – это **create dttm**, дата создания заявки (она есть у каждого человека). Вторая дата по «ранности» будет по логике – дата согласования заявки (**approve dttm**) и самая поздняя дата – дата перевода денежных средств **util dttm**.
- 2) Если у человека есть дата в колонке approve\_dttm → **approve\_flg** будет 1, иначе 0. Находим человека у которого есть approve\_dttm но нет util\_dttm, и определяем колонку approve\_flg, аналогично определяем колонку **util\_flg**.
- 3) **score** – известно, что score в диапазоне от 0 до 1, просто находим колонку, где значения лежат в этом диапазоне.

**pid** – уникальные идентификаторы, понять нетяжело (типа 5f324ac9-99de-473d-815f-b9162957d623). **Name** – колонка с ФИО, тоже предельно просто. **Age** – находим колонку где значения лежат в диапазоне от 15 до 24 → age определен. **Education level code** – уровень образования, переводим: MASTER – магистратура, BACHELOR – бакалавриат, SPECIALTY – специалитет.

**Subside rate** – ставка субсидирования, логично что это будут небольшие числа, т.к. ставка (от 3 до 5). **Gender cd** – логично, M - male(мужчина), F - female(женщина).

**Utm source** – рекламный источник, website, social\_media значения идеально подходят. **Specialty** – специальность на которой учится человек, исходя из таких значений как “прикладная математика” становится все ясно. **Reject reason** – причина отказа, по значению «заявитель найден в чс» все становится понятно.

**Short nm** – вузы, тут легко. **Semester cost amt** – цена за семестр, находим колонку где значения лежат в диапазоне от [100 000; 500 000] → сходится с реальными ценами за семестр. **Initial approved amt** – находим столбец где цифры уже очень большие (миллионы) и логично, что это и будет начальная сумма одобренного кредита. **Semestr cnt** – кол-во семестров, берем например какую-нибудь строку, берем цену за семестр у данного человека и умножаем на semestr\_cnt и получаем initial\_approved\_amt → semestr\_cnt подобран правильно. **Initial term** – срок кредитования в годах, у нас осталось только 2 колонки неопознанные, в одной из них значения только 0 и 1 → **marketing flag** нашелся, а вторая колонка значения от 0 до 6 → initial\_term. **ВСЕ столбцы опознаны верно.**

## II. Поиск формулы.

- 1) Хочется сразу разобраться со столбцами с датами (create\_dttm, util\_dttm, approve\_dttm). Я решил вытащить из них день недели: от 1 до 7, где 1 – понедельник, а 7 - воскресенье (если util\_dttm или approve\_dttm нет, то тогда я записывал значения 0). Но день недели для create\_dttm я убрал, т.к. он незначительно повышал MAE (плохо коррелировался со score).

```

df['create_dttm'] = pd.to_datetime(df['create_dttm'])
df['util_dttm'] = pd.to_datetime(df['util_dttm'])
df['approve_dttm'] = pd.to_datetime(df['approve_dttm'])
df['util_day'] = df['util_dttm'].dt.weekday + 1
df['util_day'] = df['util_day'].fillna(0)
df['approve_day'] = df['approve_dttm'].dt.weekday + 1
df['approve_day'] = df['approve_day'].fillna(0)

df['util_delay'] = (df['util_dttm'] - df['approve_dttm']).dt.days
df['util_delay'] = df['util_delay'].fillna(0)
df['util_create_delay'] = (df['util_dttm'] - df['create_dttm']).dt.days
df['util_create_delay'] = df['util_create_delay'].fillna(0)
df['approve_delay'] = (df['approve_dttm'] - df['create_dttm']).dt.days
df['approve_delay'] = df['approve_delay'].fillna(0)

```

Ниже я создал еще новые столбцы, util\_delay – сколько дней прошло между согласованием заявки и переводом денег, util\_create\_delay – сколько дней прошло между созданием заявки и переводом денег, approve\_delay – сколько дней прошло между созданием заявки и согласованием. Если какой-то даты не было, то туда я просто вставлял 0. Также добавил 2 столбца с длиной pid и длиной name (они чуток

```

df['name_length'] = df['name'].apply(len)
df['pid_length'] = df['pid'].apply(len)

```

понижали MAE, что и требовалось)

- 2) Далее я посмотрел на значения pid, и решил добавить пару новых признаков, а именно кол-во определенных букв в pid (которые я увидел в pid).

```

1 usage
def count_letters(pid):
    return {
        'count_a': pid.count('a'),
        'count_b': pid.count('b'),
        'count_c': pid.count('c'),
        'count_d': pid.count('d'),
        'count_f': pid.count('f'),
        'count_e': pid.count('e'),
        'count_all': pid.count('a') + pid.count('b') + pid.count('c') + pid.count('d') + pid.count('e') + pid.count('f')
    }

count_data = df['pid'].apply(count_letters).apply(pd.Series)
df = pd.concat([df, count_data], axis=1)

```

Score также имел зависимость, пусть и небольшую, от этих признаков.

Теперь удалим некоторые признаки от которых мы взяли то, что нам нужно было и больше они нам не нужны.

```

df.drop(columns=['approve_dttm', 'create_dttm', 'util_dttm', 'pid', 'name'], inplace=True)

```

- 3) Теперь нужно было разобраться с категориальными признаками (университет и тд). Удалять просто было не лучшим решением, можно было применить one hot encoding, но я решил заменить значения категорий на среднее значение score по определенному значению категории.

```

categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    df[col] = df[col].fillna('unknown')
    mean_scores = df.groupby(col)['score'].mean().reset_index()
    mean_scores.columns = [col, 'mean_score']
    mean_scores_dict = dict(zip(mean_scores[col], mean_scores['mean_score']))
    df[col] = df[col].map(mean_scores_dict)

```




- 4) Теперь мой датасет был проанализирован и готов к тому, чтобы применять алгоритмы машинного обучения и предсказывать score.





Я решил воспользоваться библиотекой sklearn, и взять оттуда модель линейной регрессии LinearRegressor. Сначала вычислим MAE для всей выборки, предварительно обучив модель по всей выборке, а затем еще получим MAE для тестовой выборки.

```

119 x = df.drop(columns=['score'])
120 y = df['score']
121 lin_model = LinearRegression()
122 lin_model.fit(x, y)
123 y_pred = lin_model.predict(x)
124 mae = mean_absolute_error(y, y_pred)
125 print('MAE по всей выборке:', mae)
126
127 x_train, x_test, y_train, y_test = train_test_split(*arrays: x, y, test_size=0.2, random_state=2066)
128 lin_model = LinearRegression()
129 lin_model.fit(x_train, y_train)
130 y_pred = lin_model.predict(x_test)
131 mae = mean_absolute_error(y_test, y_pred)
132 print('MAE по тестовой выборке (20 %):', mae)

```

Run   

↑ C:\Users\1\Desktop\DANO\venv\_dano\Scripts\python.exe C:\Users\1\Desktop\DANO\ii.py  
 ↓ MAE по всей выборке: 0.05188089572443557  
 ⇨ MAE по тестовой выборке (20 %): 0.05109275590446605

MAE по всей выборке вышло 0.05188 (а его и нужно было записать в ответ) и 0.0511 для тестовой выборки.

- 5) Далее я решил прибегнуть к таким мерам: я решил перебрать **все возможные комбинации признаков** и обучать на них модель, и понять: при какой комбинации MAE по всей выборке будет *минимальным*. Такой код я пошел выполнял в google colab, т.к. мой ноутбук бы никогда не завершил его. 2,5 часа я

ждал, пока мой код завершится (сам код):

```
x = df.drop(columns=['score'])
y = df['score']
best_mae = float('inf')
best_combination = None
for r in range(1, x.shape[1] + 1):
    for combination in itertools.combinations(x.columns, r):
        lin_model = LinearRegression()
        lin_model.fit(x[list(combination)], y)
        y_pred = lin_model.predict(x[list(combination)])
        mae = mean_absolute_error(y, y_pred)

        if mae < best_mae:
            print(mae, combination)
            best_mae = mae
            best_combination = combination

print(f'Лучшая комбинация: {best_combination}')
print(f'Лучший MAE: {best_mae}')
```

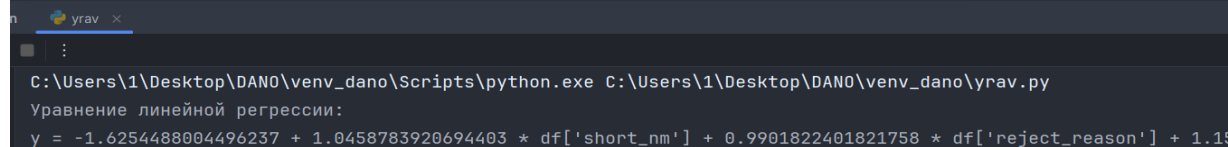
И спустя 2.5 часа я получил такой ответ:

```
0.05187872422347717 ('short_nm', 'reject_reason', 'gender_cd', 'age', 'subside_rate', 'count_all')
```

Если мы оставляем только эти 6 признаков и обучаем на них модель - то получим минимальный MAE = 0.05187 по всей выборке. MAE практически не изменился, но зато теперь наша итоговая формула содержит в себе всего лишь 6 признаков! С помощью такого кода вытаскиваем коэффициенты из модели и создаем формулу:

```
coefficients = lin_model.coef_
intercept = lin_model.intercept_
yравнение = f'y = {intercept}'
for coef, name in zip(coefficients, x.columns):
    yравнение += f" + {coef} * df['{name}']"

print("Уравнение линейной регрессии:")
print(yравнение)
```



```
yрав
C:\Users\1\Desktop\DANO\venv_dano\Scripts\python.exe C:\Users\1\Desktop\DANO\venv_dano\yрав.py
Уравнение линейной регрессии:
y = -1.6254488004496237 + 1.0458783920694403 * df['short_nm'] + 0.9901822401821758 * df['reject_reason'] + 1.1569716832089008 * df['gender_cd'] + 0.010750909048035995 * df['age'] + 0.016547518378239423 * df['subside_rate'] + -0.0037414646143657526 * df['count_all']
```

**Итоговая формула:**

```
score = -1.6254488004496237 + 1.0458783920694403 * df['short_nm'] +
0.9901822401821758 * df['reject_reason'] + 1.1569716832089008 *
df['gender_cd'] + 0.010750909048035995 * df['age'] + 0.016547518378239423 *
df['subside_rate'] + -0.0037414646143657526 * df['count_all']
```

И по итогу вот код для которого данная формула будет работать:

```

import pandas as pd
df = pd.read_csv('credit.csv') # считывание датасета

df = pd.concat([df, df['pid'].apply(lambda pid: {'count_all':
sum(pid.count(letter) for letter in 'abcdef'))).apply(pd.Series)], axis=1)
categorical_cols = ['short_nm', 'reject_reason', 'gender_cd']
for col in categorical_cols:
    df[col] = df[col].fillna('unknown')
    mean_scores = df.groupby(col)['score'].mean().reset_index()
    mean_scores.columns = [col, 'mean_score']
    df[col] = df[col].map(dict(zip(mean_scores[col],
mean_scores['mean_score'])))
# ФОРМУЛА
score = -1.6254488004496237 + 1.0458783920694403 * df['short_nm'] +
0.9901822401821758 * df['reject_reason'] + 1.1569716832089008 *
df['gender_cd'] + 0.010750909048035995 * df['age'] + 0.016547518378239423 *
df['subside_rate'] + -0.0037414646143657526 * df['count_all']

```

Выведем теперь для этой формулы MAE по всей выборке:

```

from sklearn.metrics import mean_absolute_error
print(mean_absolute_error(df['score'], score))

```

**Получаем такой MAE по всей выборке:**

**0.05187872422347752**

*Это всё, что я смог сделать. Спасибо!*