Модель склонности клиента к приобретению машиноместа

Работа выполнена студентами группы ИСП-21

Юрченко Владимир,Симбирцев Владимир,
Коряковский Артем

самолет

Ход работы

1

Определение проблемы

2

Подключение нужных для работы библиотек и работа по обработке и очистке данных

3

Написание и обучение модели

4

Подведение итогов и рефлексия

самолет

Определение проблемы

1. Ключевую роль в продажах играет эффективная целевая рассылка. Рассылки позволяют оперативно информировать клиентов об актуальных предложениях и сервисах компании. Однако каждая рассылка сопряжена с издержками:

Финансовые затраты - на подготовку и доставку сообщений (SMS, email и т.д.);
Временные затраты маркетологов и продавцов;
Риск раздражения получателей частыми сообщениями.

Это может негативно сказаться на лояльности клиентов в долгосрочной перспективе.

Подключение нужных для работы библиотек и работа по обработке и очистке данные

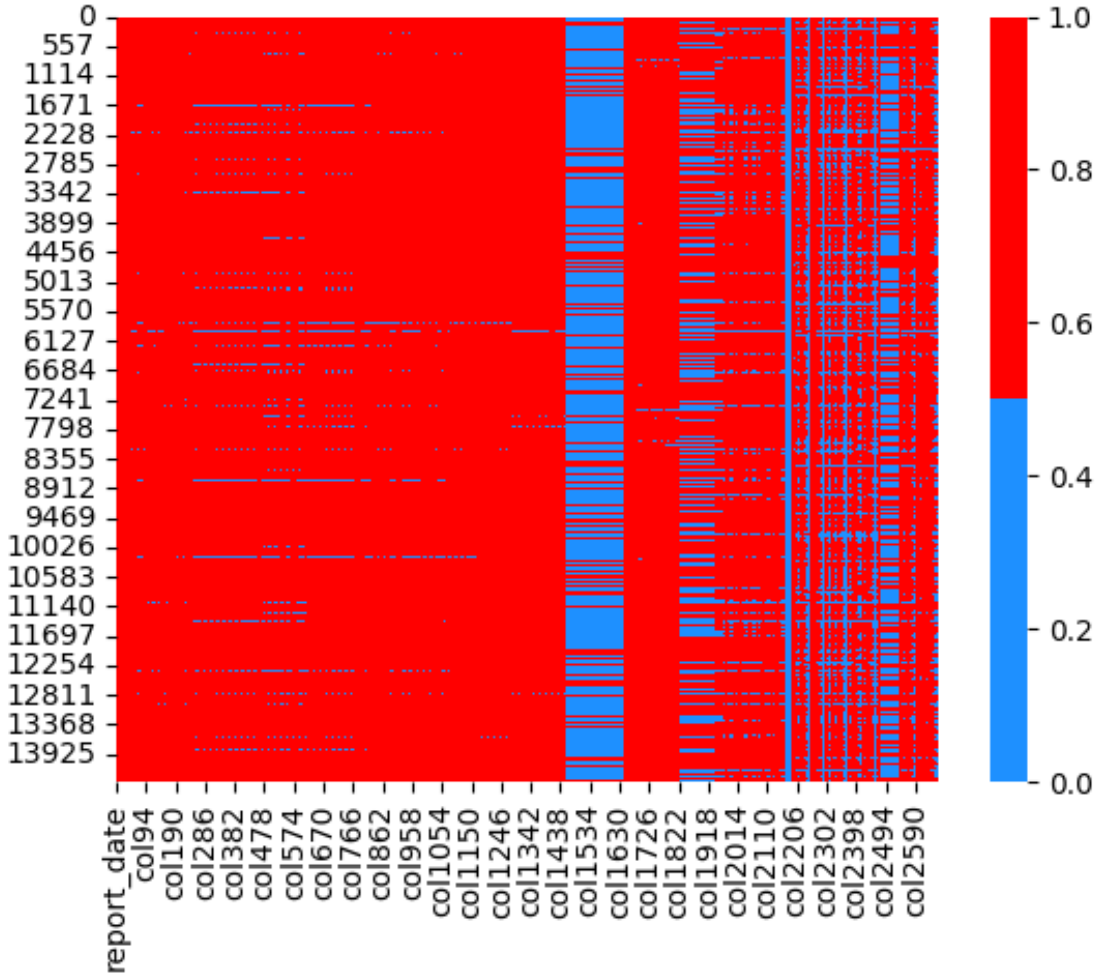
2. Для начала работы подключаем нужные библиотеки, который помогут нам в дальнейшей работе

```
import pandas as pd # Анализ данных
import numpy as np  # Поддержка многомерных массивов
import seaborn as sns
import matplotlib.pyplot as plt # Интерфейс визуализаций
from sklearn import preprocessing
```

2.1 Выводим тепловую таблицу, которая будет показывать пропуски

```
# Тепловая карта пропущенных значений
cols = df.columns # Все столбцы

# красный цвет - пропущенные данные, синий - не пропущенные
colours = ['#1E90FF', '#FF0000']
sns.heatmap(df[cols].isnull(), cmap=sns.color_palette(colours))
```

2.3 Данную тепловую карту получили



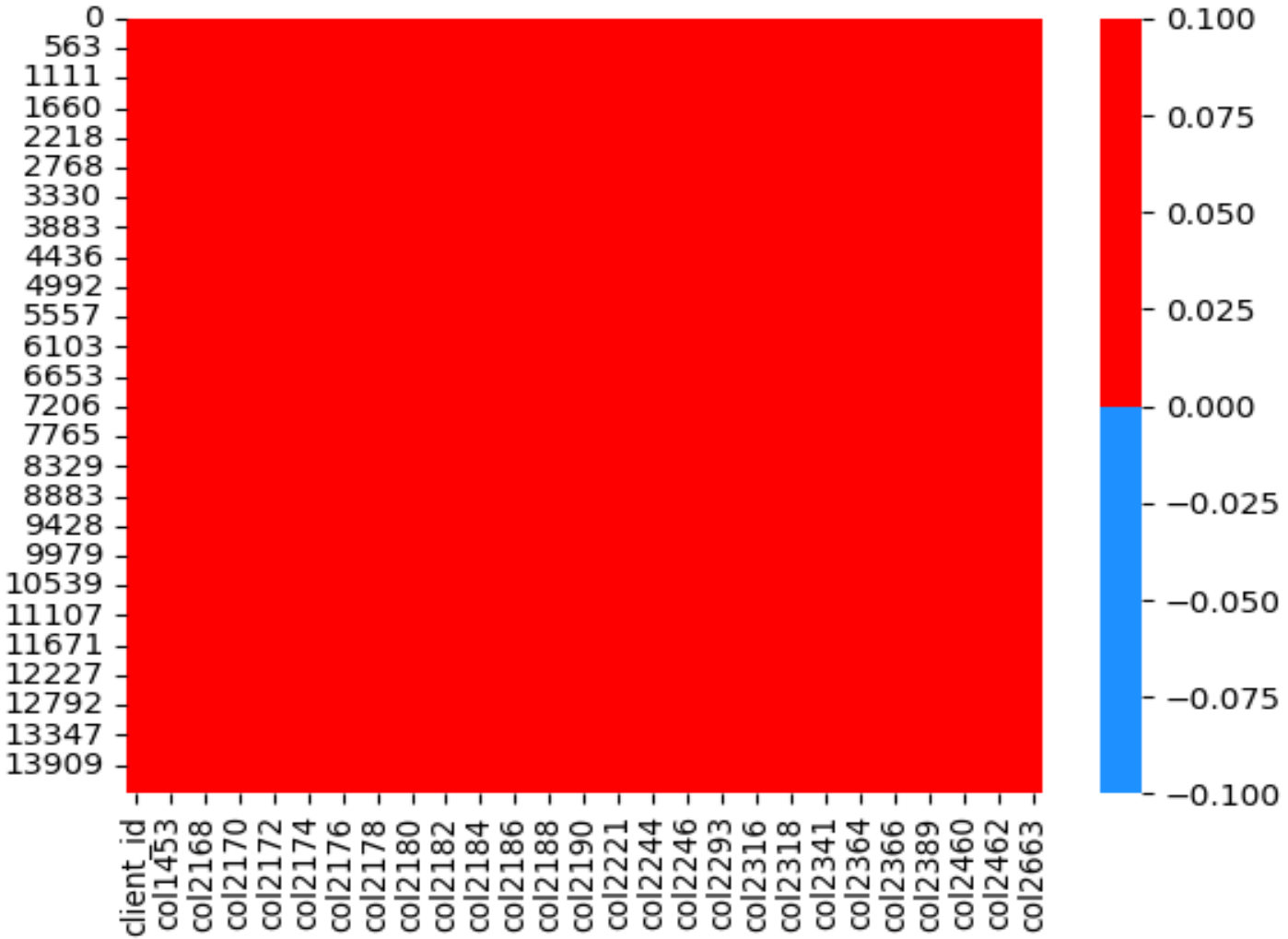
2.4 Далее чистим от пропусков

```python
# Удаление столбцов с более чем 10% пропущенных значений
threshold = 0.1 * len(df)  # 10% от общего количества строк
df = df.dropna(axis=1, thresh=len(df) - threshold)

# Удаление незаполненных строк
df = df.dropna()  # Удаляем строки, в которых есть хотя бы одно пропущенное значение

# Проверка и удаление дубликатов
df = df.drop_duplicates()  # Удаляем дубликаты
```

```python
# Удаление столбцов с типом данных object
df = df.select_dtypes(exclude=['object'])
```

```python
# Тепловая карта пропущенных значений
cols = df.columns # Все столбцы

# красный цвет - пропущенные данные, синий - не пропущенные
colours = ['#1E90FF', '#FF0000']
sns.heatmap(df[cols].isnull(), cmap=sns.color_palette(colours))
```

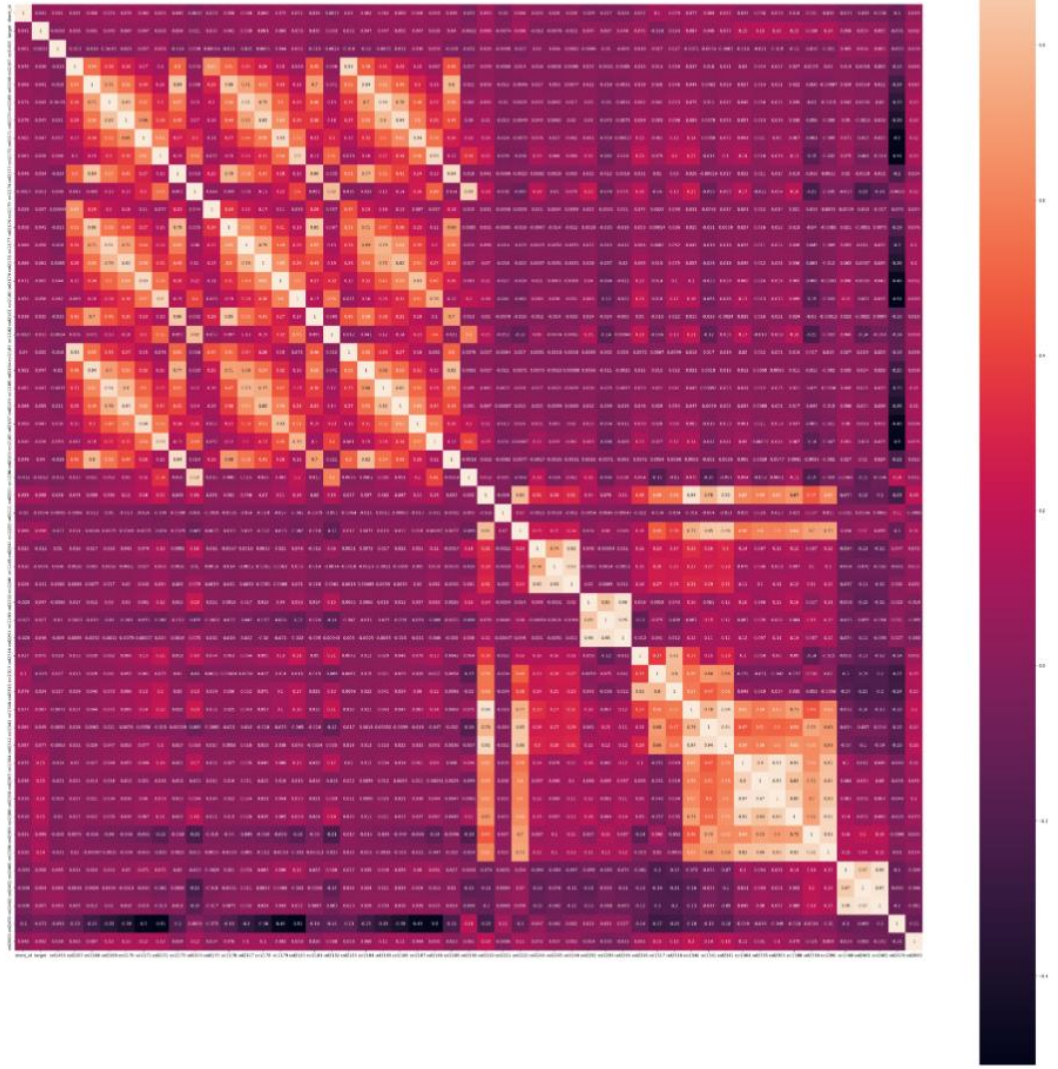
Тепловая карта после очистки от пропусков



2.5 Кодируем и выводим корреляционные матрицы

```
# Напишем функцию, которая принимает на вход наши данные, кодирует числовыми значениями
# и возвращает обновленный данные и сами кодировщики
def number_encode_features(init_df):
    result = init_df.copy() #копируем нашу исходную таблицу
    encoders = {}
    for column in result.columns:
        if result.dtypes[column] == 'object': #object -- строковый тип / если тип столбц
            encoders[column] = preprocessing.LabelEncoder() #для колонки column создаем
            result[column] = encoders[column].fit_transform(result[column]) #применяем к
    return result, encoders

encoded_data, encoders = number_encode_features(df) #Теперь encoded data содержит закоди
```

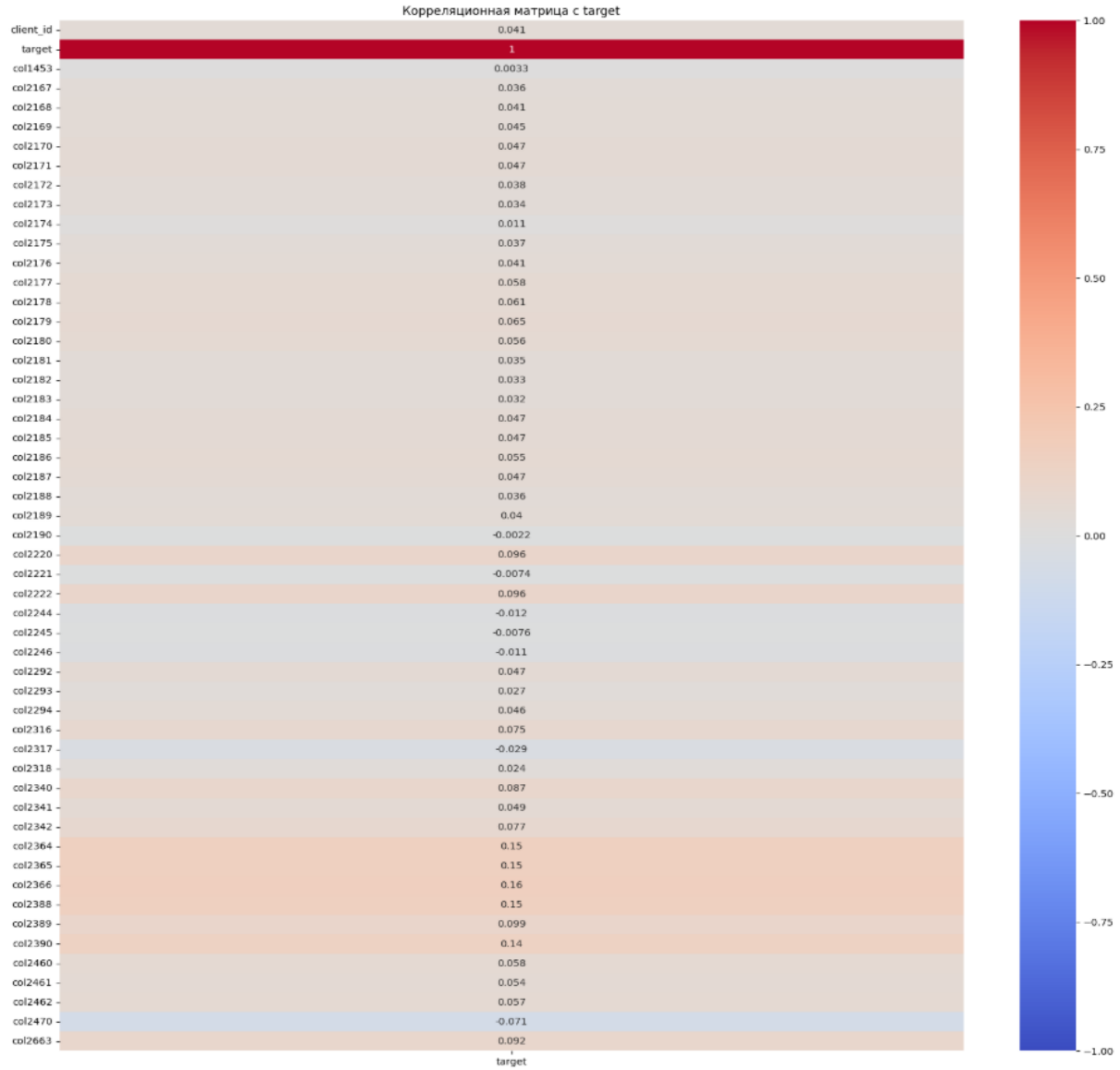
```
plt.figure(figsize = (50,50))
encoded_data, encoders = number_encode_features(df)

sns.heatmap(encoded_data.corr(), square=True, annot=True)
```



2.6 Дополнительно выводим корреляционную таблицу по target

```
encoded_data, encoders = number_encode_features(df)

# Вычисляем корреляцию
correlation_matrix = encoded_data.corr()

# Создаем тепловую карту
plt.figure(figsize=(20, 20))
sns.heatmap(correlation_matrix[['target']], annot=True, cmap='coolwarm', vmin=-1, vmax=1)

# Настраиваем заголовок
plt.title('Корреляционная матрица с target')
plt.show()
```

```
valid = pd.read_csv('valid.csv')  # Ваш датасет valid

# Получаем список колонок из train
train_columns = df.columns.tolist()

# Отбираем только те колонки из valid, которые есть в train
valid_filtered = valid[train_columns]

# Если вам нужно сохранить отфильтрованный датасет
valid_filtered.to_csv('cleanedValid.csv', index=False)
```



Написание и обучение модели

3. Добавляем нужные библиотеки

```
import pandas as pd
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, accuracy_score
```

3.1 Загружаем и подготавливаем данные

```
train = pd.read_csv('cleanedTrain.csv')
valid = pd.read_csv('cleanedValid.csv')
```

```
x_train = train.drop(columns=['target'])  # Признаки
y_train = train['target']  # Целевая переменная

x_Valid = valid.drop(columns=['target'])  # Признаки
y_Valid = valid['target']  # Целевая переменная
```

3.2 Обучаем модель

```python
# Обучение модели с гиперпараметрами
model = xgb.XGBClassifier(
    eval_metric='logloss',
    learning_rate=0.1,  # Скорость обучения
    max_depth=6,        # Максимальная глубина дерева
    n_estimators=100,   # Количество деревьев
    subsample=0.8,      # Доля выборки для обучения
    colsample_bytree=0.8,  # Доля признаков для каждого дерева
    gamma=0,            # Минимальное уменьшение потерь для разделения
    reg_alpha=0,        # L1-регуляризация
    reg_lambda=1        # L2-регуляризация
)

model.fit(x_train, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=0, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=0.1, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=6,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=100,
              n_jobs=None, num_parallel_tree=None, random_state=None, ...)
```

3.3 Предсказываем вероятность покупки

```
# Предсказание вероятности покупки
pred = model.predict_proba(x_Valid)[:, 1]  # Вероятность покупки

print("Модель обучена.")
print(roc_auc_score(y_Valid, pred))
```

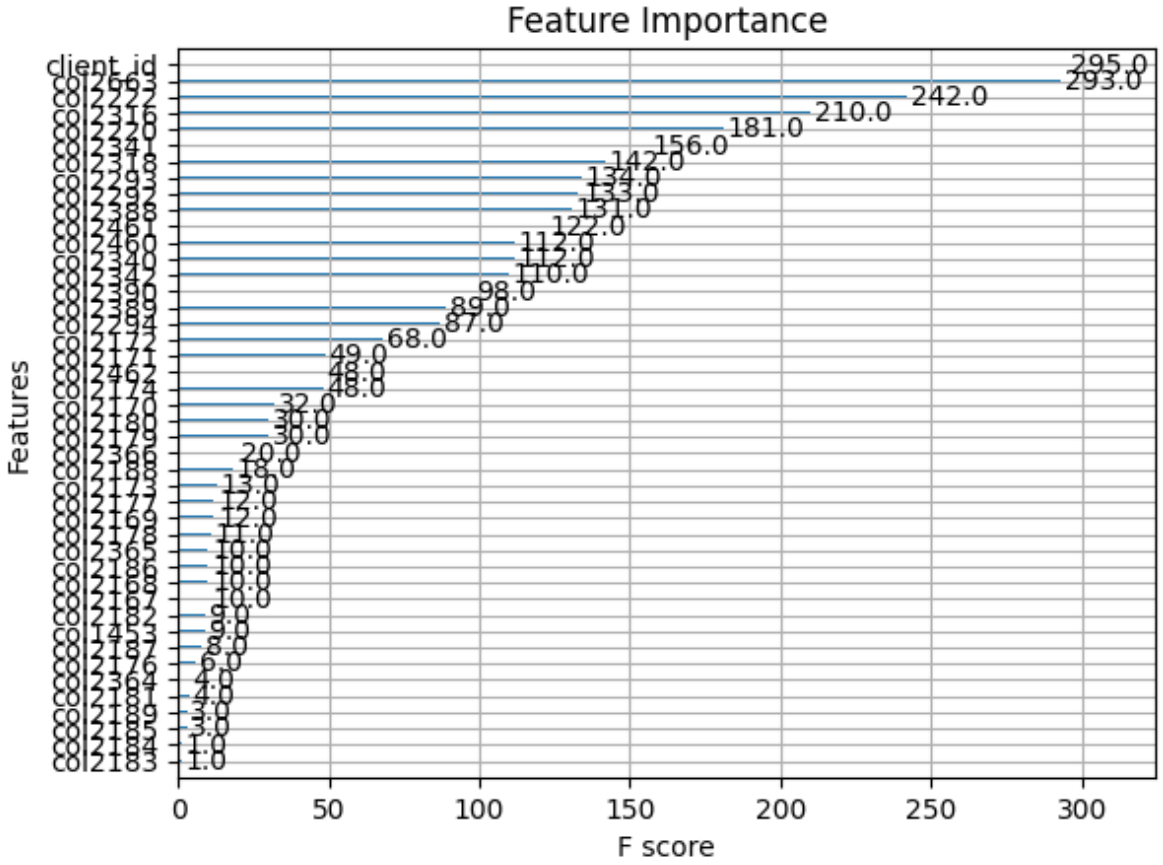
Как мы видим, модель выдает высокий результат предсказания

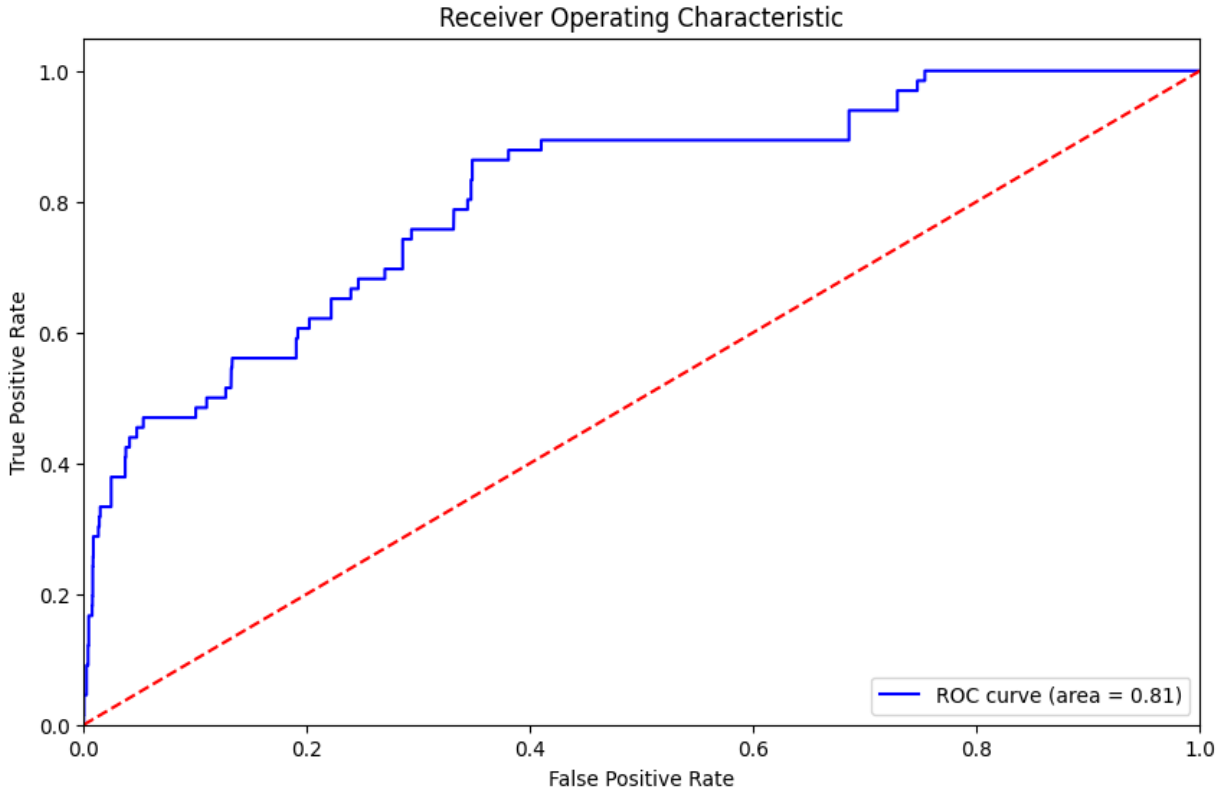
```
Модель обучена.
0.8101802420174904
```

3.4 Выводим графики

```
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt

fpr, tpr, thresholds = roc_curve(y_Valid, pred)
plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='blue', label='ROC curve (area = %0.2f)' % roc_auc_score(y_Valid, pred))
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.show()

# 9. Визуализация важности признаков
plt.figure(figsize=(10, 6))
xgb.plot_importance(model, importance_type='weight')
plt.title('Feature Importance')
plt.show()
```

Выводим дополнительные графики





Подведение итогов и рефлексия

Мы успешно провели грамотную очистку данных, что привело к высокой точности предсказаний нашей модели. В процессе работы мы освоили создание модели, которая демонстрирует высокую эффективность в выполнении заданной функциональности.

Однако есть возможности для дальнейшего улучшения:

-Добавление дополнительных данных может значительно обогатить модель и повысить ее предсказательную силу.

-Также можно рассмотреть оптимизацию существующих признаков, что позволит улучшить качество результатов.