Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г. В. Плеханова»

Кафедра информатики

Выпускная квалификационная работа

по программе профессиональной переподготовке «Процедурно-ориентированное программирование в прикладных задачах анализа данных в экономике»

на тему «Прогноз оттока клиентов банка по продукту "Кредитные карты" »

Выполнил:

Кудрявцева Евгения Родионовна

Преподаватель:

Красиков Виталий Александрович

Москва

2022

1. Понимание бизнес-целей

1.1. Понимание бизнеса

Возрастает конкуренция среди банков по продаже услуг «кредитная карта».
Для решения данной задачи требуется выполнить следующие исследования:

- Определить и визуализировать, какие факторы способствуют оттоку клиентов;

- Создать модель прогнозирования, которая будет выполнять следующие действия: определить, собирается ли клиент уйти или нет

Ожидаемые преимущества: облегчить службе поддержки клиентов поиск неудовлетворенных клиентов в их усилиях по предотвращению оттока.

1.2. Доступные ресурсы

Для успешной реализации проекта необходимы следующие категории специалистов: аналитик данных, бизнес-аналитик, специалист по базам данных, руководитель проекта.
Заказчик располагает всем необходимым оборудованием для поведения анализа данных.

1.3. Риски

Риск нехватки и неполноты данных
Риск несоответствия полученных результатов требованиям заказчика.

1.4. Ограничения

Ограничение сроков: 1 месяц.
Ставки по сотрудникам: Аналитик данных – 1 ставка.

1.5. Цели исследования данных

Мы стремимся выполнить следующее для этого исследования:

1) Определить и визуализировать, какие факторы способствуют оттоку клиентов;

2) Создать модель прогнозирования, которая будет выполнять следующие действия:

-Определить, собирается ли клиент уйти или нет;

- Выбрать модель, которая будет связывать вероятность с оттоком, чтобы облегчить службе поддержки клиентов поиск клиентов, которые могут уйти, в их усилиях по предотвращению оттока.

1.6.Критерии успешности изучения данных.

Метрики оценки точности и качества построенных моделей:
Для моделей регрессии: качество модели определяется с использованием коэффициента детерминации (R2), точность модели определяется на основании средней относительной ошибки (MAPE).
Границы значений метрик: R2 должен быть больше либо равен 0.8, MAPE не более 10%.
Метрики оценки качества классификации: accuracy, ROC-кривая, таблицы сопряженности.

2. Начальное изучение данных.

2.1.Сбор данных.

2.2.Описание данных

Объем данных – 1000 строк с 14 атрибутами
нет пропущенных значений
Типы, виды данных и схемы кодирования

Наименование	Тип данных	Вид данных	Схема кодирования
CreditScore Кредитный рейтинг	Целое число	Непрерывный	-
Geography	Строковый	Дискретный	Целое число
Gender	Строковый	Дискретный	Целое число
Age	Целое число	Непрерывный	-
Tenure Срок пребывания в должности	Целое число	Непрерывный	-
Balance	Число с плавающей запятой	Непрерывный	-
NumOfProducts Количество продуктов	Целое число	Непрерывный	-
HasCrCard Есть карта или нет (да/нет)	Целое число	Непрерывный	-
IsActiveMember Активный участник	Целое число	Непрерывный	-

(да/нет)			
EstimatedSalary зарплата	Число с плавающей запятой	Непрерывный	-
Exited Клиент ушел (вышел) (да/нет)	Целое число	Непрерывный	-

Формат данных – файл csv, разделитель – “,”.
Из вышеизложенного осталась пара вопросов:

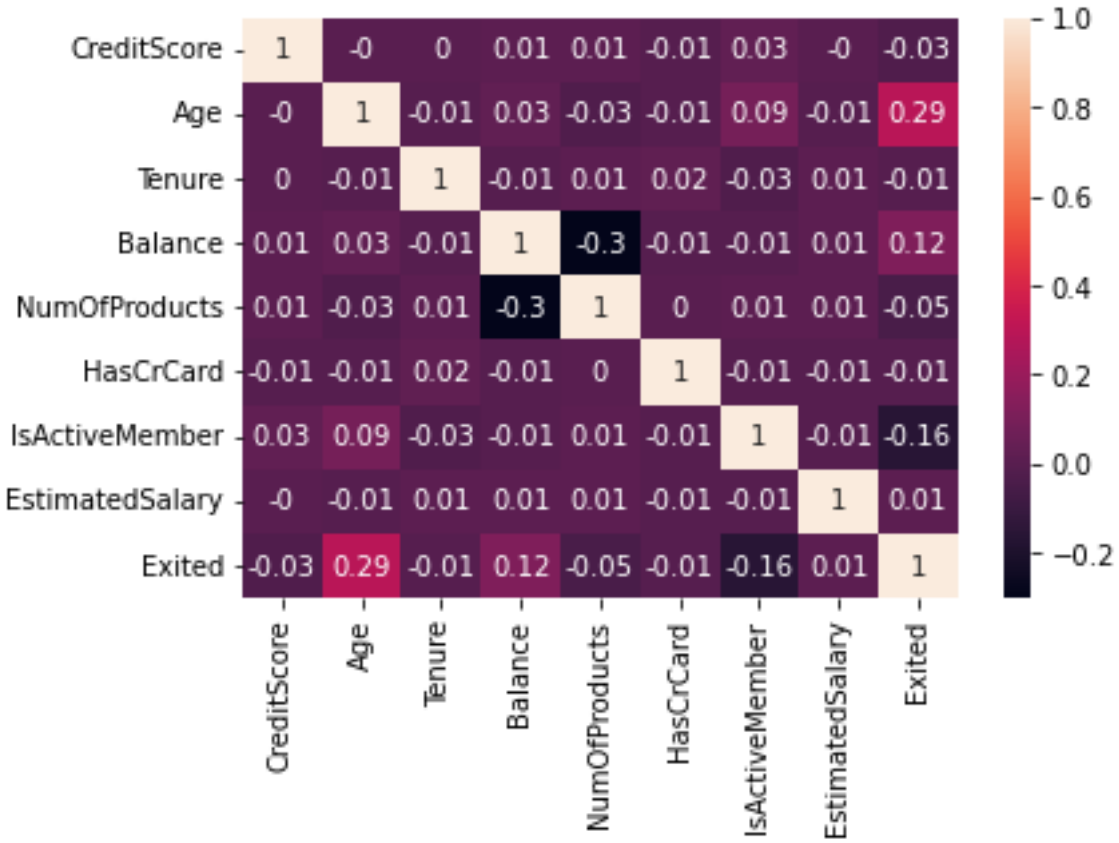
Данные представляются снимком в какой-то момент времени, например. баланс на заданную дату, что оставляет много вопросов:
Какая сегодня дата и какое значение имеет эта дата
Можно ли получить балансы за определенный период времени, а не за одну дату.
Есть клиенты, которые вышли, но все еще имеют баланс на своем счете! Что бы это значило? Могли ли они выйти из продукта, а не из банка?
Что значит быть активным членом и есть ли в этом разница? Может быть, вместо этого лучше указать количество транзакций как по кредитам, так и по дебетам на счете?
Разбивка продуктов, купленных покупателем, может предоставить больше информации, чем список количества продуктов.

Таким образом, у нас в основном есть категориальные переменные и 5 непрерывных переменных.

2.3.Исследование данных

Здесь наш главный интерес состоит в том, чтобы получить представление о том, как данные атрибуты соотносятся со статусом «Выход» (отток клиентов, клиент ушел).

Построение описательной статистики



Экстремальных значений в исследуемых данных не обнаружено. Предобработка не требуется.

Проверка пропущенных значений

CustomerId 0
Surname 0
CreditScore 0
Geography 0
Gender 0
Age 0
Tenure 0
Balance 0
NumOfProducts 0
HasCrCard 0
IsActiveMember 0
EstimatedSalary 0
Exited 0
Пропущенные значения отсутствуют.

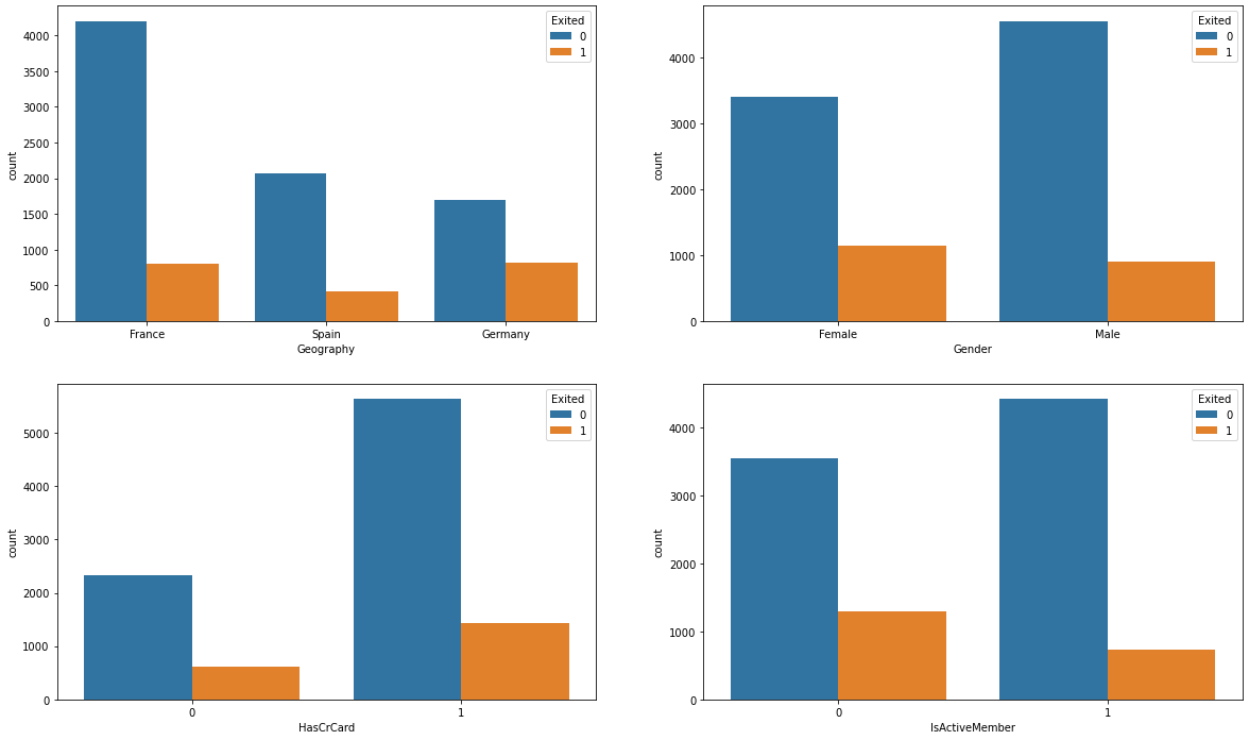
3. Подготовка данных

Данные не требуют очистки, так как не выявлено дубликатов, пропусков, противоречий и экстремальных значений.

Однако нам не потребуются первые 2 атрибута, поскольку они специфичны для клиента. ID клиента и фамилия, нам не нужна привязка к конкретному клиенту, поэтому мы исключаем и это.

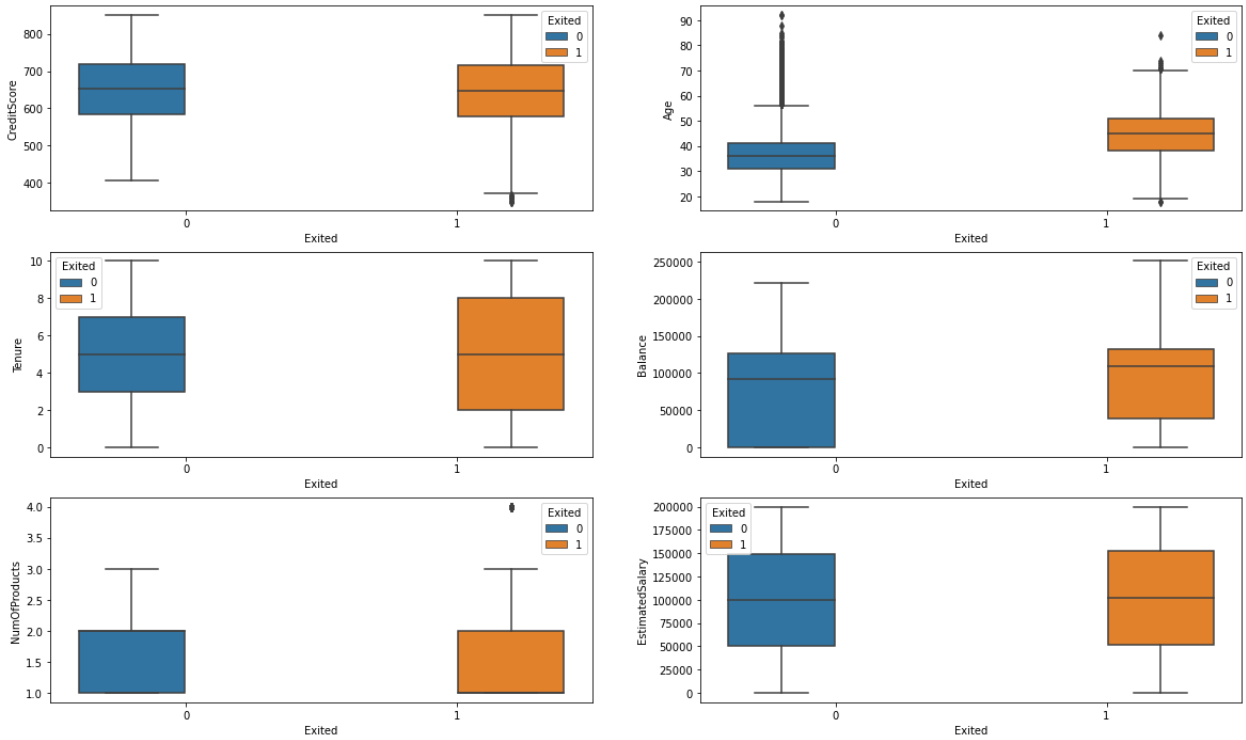
4. **Моделирование**

Видно, что 20% клиентов ушли. Таким образом, базовая модель может предсказать, что 20% клиентов уйдут. Учитывая, что 20% — это небольшое число, нам нужно убедиться, что выбранная модель действительно предсказывает с большой точностью эти 20%, поскольку банк заинтересован в выявлении и сохранении этой группы, а не в точном прогнозировании удержанных клиентов.



Отметим следующее:

- Большинство данных получены от лиц из Франции. Тем не менее, **доля ушедших клиентов обратно пропорциональна количеству клиентов**, что указывает на то, что у банка, возможно, есть проблема (возможно, недостаточно ресурсов, выделенных для обслуживания клиентов).
- Доля оттока клиентов-**женщин ниже**, чем доля клиентов-мужчин.
- Интересно, что большинство клиентов, которые ушли, **это клиенты с кредитными картами**. Учитывая, что у большинства клиентов есть кредитные карты, это может оказаться просто совпадением.
- Неудивительно, что **у неактивных членов отток больше**. Беспокоит то, что общая доля неактивных клиентов довольно высока, что говорит о том, что банку может понадобиться программа, чтобы превратить эту группу в активных клиентов, поскольку это определенно окажет положительное влияние на отток клиентов.



Отметим следующее:

Пожилые клиенты обращают на себя больше внимания (отток выше), чем молодые, что намекает на разницу в предпочтениях в обслуживании в возрастных категориях. Банку может потребоваться пересмотреть свой целевой рынок или пересмотреть стратегию удержания между различными возрастными группами.
Что касается **стажа работы**, клиенты, находящиеся на крайнем конце (очень мало либо наоборот много), с большей вероятностью уйдут по сравнению с теми, кто имеет средний стаж работы..
К сожалению, банк теряет клиентов **со значительными остатками** на банковских счетах, что, вероятно, ударит по их доступному капиталу для кредитования.
Ни продукт, ни зарплата не имеют существенного влияния на вероятность оттока.

я попробую следующие модели:

- Логистическая регрессия
- Полиномиальная регрессия
- Случайный лес

5. Оценка результатов

Исходя из приведенных выше результатов, моя главная цель состоит в том, чтобы предсказать клиентов, которые, возможно, уйдут, чтобы их можно было включить в какую-то схему для предотвращения оттока. модель.
Учитывая, что в данных у нас было только 20% оттока, отзыв, превышающий этот базовый уровень, уже будет улучшением, но мы хотим получить как можно больше, пытаясь поддерживать высокую точность, чтобы банк мог эффективно обучать свои ресурсы для клиентов, выделенных моделью, не тратя слишком много ресурсов на ложные срабатывания.
Из приведенного выше обзора подогнанных моделей лучшей моделью, обеспечивающей достойный баланс полноты и точности, является случайный лес, где, согласно подгонке обучающей выборки, с оценкой точности 1 из 0,88, из всех клиентов, которые модель думает, что уйдут, 88% действительно уйдут, а с показателем отзыва 0,53 по единицам модель способна выделить 53% всех тех, кто ушёл.

Заключение
Точность модели на ранее неизвестных тестовых данных немного выше в отношении прогнозирования 1, то есть тех клиентов, которые уходят. Однако, несмотря на то, что модель имеет высокую точность, она все же пропускает около половины тех, кто в конечном итоге уходит. Это можно было бы улучшить, обеспечив переобучение модели с большим количеством данных с течением времени.

6. Внедрение.

Приложение 1. Код программы Python для проведенного анализа

```
-*- coding: utf-8 -*-
"""
Created on Wed Dec 09 21:26:49 2022

@author: jane9
"""

## REQUIRED LIBRARIES
# For data wrangling
import numpy as np
import pandas as pd

# For visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
pd.options.display.max_rows = None
pd.options.display.max_columns = None



# Read the data frame
# df = pd.read_csv('../input/Churn_Modelling.csv', delimiter=',')
df = pd.read_csv('Churn_Modelling.csv', delimiter=',')
df.shape
print('Объем данных: ',df.shape)

# Проверка пропущенных значений
df.isnull().sum()
print('Проверка пропущенных значений: ', df.isnull().sum() )

# Получить уникальный счетчик для каждой переменной
df.nunique()
print('Получить уникальный счетчик для каждой переменной:',df.nunique())

# Отбросим два столбца, как описано выше. нам не потребуются первые 2 атрибута,
поскольку они специфичны для клиента.
# это столюцы CustomerId и Surname
df = df.drop(["RowNumber", "CustomerId", "Surname"], axis = 1)
# Review the top rows of what is left of the data frame
df.head()



# Проверить типы переменных данныхdf.dtypes
print ('Типы данных:', df.dtypes)

print ('Здесь наш главный интерес состоит в том, чтобы получить представление о том, ')
print ('как данные атрибуты соотносятся со статусом «Выход».')
labels = 'Ушли', 'Остались'
sizes = [df.Exited[df['Exited']==1].count(), df.Exited[df['Exited']==0].count()]
explode = (0, 0.1)
fig1, ax1 = plt.subplots(figsize=(10, 8))
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')
plt.title("Доля клиентов, которые ушли и тех, которые остались", size = 20)
plt.show()

print(sns.heatmap(round(df.corr(), 2), annot = True))

# Сначала мы рассмотрим отношение «Статус» с категориальными переменными.
fig, axarr = plt.subplots(2, 2, figsize=(20, 12))
```

```
sns.countplot(x='Geography', hue = 'Exited',data = df, ax=axarr[0][0])
sns.countplot(x='Gender', hue = 'Exited',data = df, ax=axarr[0][1])
sns.countplot(x='HasCrCard', hue = 'Exited',data = df, ax=axarr[1][0])
sns.countplot(x='IsActiveMember', hue = 'Exited',data = df, ax=axarr[1][1])

# Отношения, основанные на непрерывных атрибутах данных
fig, axarr = plt.subplots(3, 2, figsize=(20, 12))
sns.boxplot(y='CreditScore',x = 'Exited', hue = 'Exited',data = df, ax=axarr[0][0])
sns.boxplot(y='Age',x = 'Exited', hue = 'Exited',data = df , ax=axarr[0][1])
sns.boxplot(y='Tenure',x = 'Exited', hue = 'Exited',data = df, ax=axarr[1][0])
sns.boxplot(y='Balance',x = 'Exited', hue = 'Exited',data = df, ax=axarr[1][1])
sns.boxplot(y='NumOfProducts',x = 'Exited', hue = 'Exited',data = df, ax=axarr[2][0])
sns.boxplot(y='EstimatedSalary',x = 'Exited', hue = 'Exited',data = df, ax=axarr[2]  [1])

# раздел 4. Мы стремимся добавить функции, которые могут повлиять на вероятность оттока.
# Сначала мы разделяем train и тестовые наборы
print ('Разделим train и тестовые наборы:')
# Split Train, test data
df_train = df.sample(frac=0.8,random_state=200)
df_test = df.drop(df_train.index)
print(len(df_train))
print(len(df_test))

df_train['BalanceSalaryRatio'] = df_train.Balance/df_train.EstimatedSalary
sns.boxplot(y='BalanceSalaryRatio',x = 'Exited', hue = 'Exited',data = df_train)
plt.ylim(-1, 5)

# получили картинку

# Учитывая, что срок пребывания в должности является «функцией» возраста, мы вводим переменную
# с целью стандартизировать срок пребывания в должности по возрасту:
print ('Учитывая, что срок пребывания в должности является «функцией» возраста,')
print ('мы вводим переменную с целью стандартизировать срок пребывания в должности по возрасту:')
df_train['TenureByAge'] = df_train.Tenure/(df_train.Age)
sns.boxplot(y='TenureByAge',x = 'Exited', hue = 'Exited',data = df_train)
plt.ylim(-1, 1)
plt.show()

# вводим переменную для определения кредитного рейтинга с учетом возраста, чтобы учесть кредитное поведение во взрослой жизни
print('вводим переменную для определения кредитного рейтинга с учетом возраста.')
df_train['CreditScoreGivenAge'] = df_train.CreditScore/(df_train.Age)
# Resulting Data Frame
```

```
df_train.head()
print (df_train.head())

# Подготовка данных для подбора модели

# Упорядочим столбцы по типу данных для упрощения
print('Упорядочим столбцы по типу данных для упрощения.')

continuous_vars = ['CreditScore',  'Age', 'Tenure', 'Balance','NumOfProducts', 'EstimatedSalary',
'BalanceSalaryRatio', 'TenureByAge','CreditScoreGivenAge']
# KeyError: "['TenureByAge'] not in index"   ?????????????
cat_vars = ['HasCrCard', 'IsActiveMember','Geography', 'Gender']
df_train = df_train[['Exited'] + continuous_vars + cat_vars]
df_train.head()

# Для одной неподходящей переменной мы меняем 0 на -1, чтобы модели могли
фиксировать отрицательное отношение
# где атрибут inapplicable вместо 0 '"
print ('Для одной неподходящей переменной мы меняем 0 на -1, где атрибут inapplicable
вместо 0.')
df_train.loc[df_train.HasCrCard == 0, 'HasCrCard'] = -1
df_train.loc[df_train.IsActiveMember == 0, 'IsActiveMember'] = -1
df_train.head()

# кодирование категориальных переменных
print('Делаем кодирование категориальных переменных.')
lst = ['Geography', 'Gender']
remove = list()
for i in lst:
    if (df_train[i].dtype == np.str or df_train[i].dtype == np.object):
        for j in df_train[i].unique():
            df_train[i+'_'+j] = np.where(df_train[i] == j,1,-1)
        remove.append(i)
df_train = df_train.drop(remove, axis=1)
df_train.head()

# minMax масштабирование непрерывных переменных
print ('minMax масштабирование непрерывных переменных')

minVec = df_train[continuous_vars].min().copy()
maxVec = df_train[continuous_vars].max().copy()
df_train[continuous_vars] = (df_train[continuous_vars]-minVec)/(maxVec-minVec)
df_train.head()

# конвейер подготовки данных для тестовых данных
print ('Подготовка тестовых данных.')
```

```
def DfPrepPipeline(df_predict,df_train_Cols,minVec,maxVec):
    # Add new features
    df_predict['BalanceSalaryRatio'] = df_predict.Balance/df_predict.EstimatedSalary
    df_predict['TenureByAge'] = df_predict.Tenure/(df_predict.Age - 18)
    df_predict['CreditScoreGivenAge'] = df_predict.CreditScore/(df_predict.Age - 18)


    # Reorder the columns
    continuous_vars =
['CreditScore','Age','Tenure','Balance','NumOfProducts','EstimatedSalary','BalanceSalaryRatio','T
enureByAge','CreditScoreGivenAge']

    cat_vars = ['HasCrCard','IsActiveMember',"Geography", "Gender"]
    df_predict = df_predict[['Exited'] + continuous_vars + cat_vars]
    # Change the 0 in categorical variables to -1
    df_predict.loc[df_predict.HasCrCard == 0, 'HasCrCard'] = -1
    df_predict.loc[df_predict.IsActiveMember == 0, 'IsActiveMember'] = -1
    # One hot encode the categorical variables
    lst = ["Geography", "Gender"]
    remove = list()
    for i in lst:
        for j in df_predict[i].unique():
            df_predict[i+'_'+j] = np.where(df_predict[i] == j,1,-1)
        remove.append(i)
    df_predict = df_predict.drop(remove, axis=1)
    # Ensure that all one hot encoded variables that appear in the train data appear in the
subsequent data
    L = list(set(df_train_Cols) - set(df_predict.columns))
    for l in L:
        df_predict[str(l)] = -1
    # MinMax scaling coontinuous variables based on min and max from the train data
    df_predict[continuous_vars] = (df_predict[continuous_vars]-minVec)/(maxVec-minVec)
    # Ensure that The variables are ordered in the same way as was ordered in the train set
    df_predict = df_predict[df_train_Cols]
    return df_predict


# Для примерки модели я попробую следующее

# Логистическая регрессия в простом пространстве и с разными ядрами
# SVM в исходном виде и с разными ядрами
# Модели ансамбля
#  ОООЧЕНЬ ДОЛГО РАБОТАЕТ
# !Предупреждение. Этот раздел занимает ооооочень много времени, поэтому у вас есть
возможность перейти
# к следующему разделу, где я подбираю лучшие модели из этого раздела.
```

```
# Support functions
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from scipy.stats import uniform

# Fit models
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Scoring functions
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

# Function to give best model score and parameters
def best_model(model):
    print(model.best_score_)
    print(model.best_params_)
    print(model.best_estimator_)
def get_auc_scores(y_actual, method,method2):
    auc_score = roc_auc_score(y_actual, method);
    fpr_df, tpr_df, _ = roc_curve(y_actual, method2);
    return (auc_score, fpr_df, tpr_df)



# Пробую модель Логистическа регрессия
print ('Логистическа регрессия.')
param_grid = {'C': [0.1,0.5,1,10,50,100], 'max_iter': [250],
'fit_intercept':[True],'intercept_scaling':[1],
              'penalty':['l2'], 'tol':[0.00001,0.0001,0.000001]}
log_primal_Grid = GridSearchCV(LogisticRegression(solver='lbfgs'),param_grid, cv=10,
refit=True, verbose=0)
log_primal_Grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(log_primal_Grid)



# Пробую модель полиномиаль Логистическа регрессия
print('Подберем логистическую регрессию с полиномиальным ядром степени 2')
param_grid = {'C': [0.1,10,50], 'max_iter': [300,500],
'fit_intercept':[True],'intercept_scaling':[1],'penalty':['l2'],
```

```
            'tol':[0.0001,0.000001]}
poly2 = PolynomialFeatures(degree=2)
df_train_pol2 = poly2.fit_transform(df_train.loc[:, df_train.columns != 'Exited'])
log_pol2_Grid = GridSearchCV(LogisticRegression(solver = 'liblinear'),param_grid, cv=5,
refit=True, verbose=0)
log_pol2_Grid.fit(df_train_pol2,df_train.Exited)
best_model(log_pol2_Grid)


# Fit SVM with RBF Kernel
print ('Установим SVM с ядром  RBF Kernel')
param_grid = {'C': [0.5,100,150], 'gamma': [0.1,0.01,0.001],'probability':[True],'kernel': ['rbf']}
SVM_grid = GridSearchCV(SVC(), param_grid, cv=3, refit=True, verbose=0)
SVM_grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(SVM_grid)

# Fit SVM with pol kernel
print ('Установим SVM с pol kernel')
param_grid = {'C': [0.5,1,10,50,100], 'gamma': [0.1,0.01,0.001],'probability':[True],'kernel':
['poly'],'degree':[2,3] }
SVM_grid = GridSearchCV(SVC(), param_grid, cv=3, refit=True, verbose=0)
SVM_grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(SVM_grid)

# Fit random forest classifier
print('Установим случайный лес.')
param_grid = {'max_depth': [3, 5, 6, 7, 8], 'max_features':
[2,4,6,7,8,9],'n_estimators':[50,100],'min_samples_split': [3, 5, 6, 7]}
RanFor_grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, refit=True,
verbose=0)
RanFor_grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(RanFor_grid)

# Fit Extreme Gradient boosting classifier
print ('Установим Extreme Gradient boosting classifier')
param_grid = {'max_depth': [5,6,7,8], 'gamma': [0.01,0.001,0.001],'min_child_weight':[1,5,10],
'learning_rate': [0.05,0.1, 0.2, 0.3], 'n_estimators':[5,10,20,100]}
xgb_grid = GridSearchCV(XGBClassifier(), param_grid, cv=5, refit=True, verbose=0)
xgb_grid.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
best_model(xgb_grid)



# Fit best Models
print('Лучшая модель.')
```

```
# Fit primal logistic regression
print ('Попробуем primal logistic regression.')
log_primal = LogisticRegression(C=100, class_weight=None, dual=False,
fit_intercept=True,intercept_scaling=1, max_iter=250, multi_class='warn',n_jobs=None,
                    penalty='l2', random_state=None, solver='lbfgs',tol=1e-05, verbose=0,
warm_start=False)
log_primal.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)

# Fit logistic regression with pol 2 kernel
print('Попробуем logistic regression with pol 2 kernel.')
poly2 = PolynomialFeatures(degree=2)
df_train_pol2 = poly2.fit_transform(df_train.loc[:, df_train.columns != 'Exited'])
log_pol2 = LogisticRegression(C=10, class_weight=None, dual=False,
fit_intercept=True,intercept_scaling=1, max_iter=300, multi_class='warn', n_jobs=None,
                    penalty='l2', random_state=None, solver='liblinear',tol=0.0001, verbose=0,
warm_start=False)
log_pol2.fit(df_train_pol2,df_train.Exited)

# Fit SVM with RBF Kernel
print('Попробуем SVM with RBF Kernel.')
SVM_RBF = SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf', max_iter=-1,
probability=True,
        random_state=None, shrinking=True,tol=0.001, verbose=False)
SVM_RBF.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)


# Fit SVM with Pol Kernel
print ('Попробуем SVM with Pol Kernel.')
SVM_POL = SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=2, gamma=0.1, kernel='poly',  max_iter=-1,
        probability=True, random_state=None, shrinking=True, tol=0.001, verbose=False)
SVM_POL.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)


# Fit Random Forest classifier
print('Попробуем Random Forest classifier.')
RF = RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',max_depth=8, max_features=6,
max_leaf_nodes=None,min_impurity_decrease=0.0,
                    min_impurity_split=None,min_samples_leaf=1,
min_samples_split=3,min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=None,
                    oob_score=False, random_state=None, verbose=0,warm_start=False)
RF.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)
```

```
# Fit Extreme Gradient Boost Classifier
print('Попробуем Extreme Gradient Boost Classifier')
XGB = XGBClassifier(base_score=0.5, booster='gbtree',
colsample_bylevel=1,colsample_bytree=1, gamma=0.01, learning_rate=0.1,
max_delta_step=0,max_depth=7,
                    min_child_weight=5, missing=None, n_estimators=20,n_jobs=1, nthread=None,
objective='binary:logistic', random_state=0,reg_alpha=0,
                    reg_lambda=1, scale_pos_weight=1, seed=None, silent=True, subsample=1)
XGB.fit(df_train.loc[:, df_train.columns != 'Exited'],df_train.Exited)


# Проверим точность соответствия модели: большой интерес вызывает
производительность в прогнозировании 1 (клиенты, которые уходят)

print ('Проверим точность соответствия модели: большой интерес вызывает
производительность в прогнозировании 1 (клиенты, которые уходят)')

print(classification_report(df_train.Exited, log_primal.predict(df_train.loc[:, df_train.columns !=
'Exited'])))


print(classification_report(df_train.Exited,  log_pol2.predict(df_train_pol2)))


print(classification_report(df_train.Exited,  SVM_RBF.predict(df_train.loc[:, df_train.columns
!= 'Exited'])))


print(classification_report(df_train.Exited,  SVM_POL.predict(df_train.loc[:, df_train.columns
!= 'Exited'])))


print(classification_report(df_train.Exited,  RF.predict(df_train.loc[:, df_train.columns !=
'Exited'])))

print(classification_report(df_train.Exited,  XGB.predict(df_train.loc[:, df_train.columns !=
'Exited'])))

y = df_train.Exited
X = df_train.loc[:, df_train.columns != 'Exited']
X_pol2 = df_train_pol2
auc_log_primal, fpr_log_primal, tpr_log_primal = get_auc_scores(y,
log_primal.predict(X),log_primal.predict_proba(X)[:,1])
auc_log_pol2, fpr_log_pol2, tpr_log_pol2 = get_auc_scores(y,
log_pol2.predict(X_pol2),log_pol2.predict_proba(X_pol2)[:,1])
```

```
auc_SVM_RBF, fpr_SVM_RBF, tpr_SVM_RBF = get_auc_scores(y,
SVM_RBF.predict(X),SVM_RBF.predict_proba(X)[:,1])
auc_SVM_POL, fpr_SVM_POL, tpr_SVM_POL = get_auc_scores(y,
SVM_POL.predict(X),SVM_POL.predict_proba(X)[:,1])
auc_RF, fpr_RF, tpr_RF = get_auc_scores(y, RF.predict(X),RF.predict_proba(X)[:,1])
auc_XGB, fpr_XGB, tpr_XGB = get_auc_scores(y,
XGB.predict(X),XGB.predict_proba(X)[:,1])



plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_log_primal, tpr_log_primal, label = 'log primal Score: ' + str(round(auc_log_primal,
5)))
plt.plot(fpr_log_pol2, tpr_log_pol2, label = 'log pol2 score: ' + str(round(auc_log_pol2, 5)))
plt.plot(fpr_SVM_RBF, tpr_SVM_RBF, label = 'SVM RBF Score: ' +
str(round(auc_SVM_RBF, 5)))
plt.plot(fpr_SVM_POL, tpr_SVM_POL, label = 'SVM POL Score: ' +
str(round(auc_SVM_POL, 5)))
plt.plot(fpr_RF, tpr_RF, label = 'RF score: ' + str(round(auc_RF, 5)))
plt.plot(fpr_XGB, tpr_XGB, label = 'XGB score: ' + str(round(auc_XGB, 5)))
plt.plot([0,1], [0,1], 'k--', label = 'Random: 0.5')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
#plt.savefig('roc_results_ratios.png')
plt.show()


# Make the data transformation for test data
df_test = DfPrepPipeline(df_test,df_train.columns,minVec,maxVec)
df_test = df_test.mask(np.isinf(df_test))
df_test = df_test.dropna()
df_test.shape

print(classification_report(df_test.Exited,  RF.predict(df_test.loc[:, df_test.columns !=
'Exited'])))

auc_RF_test, fpr_RF_test, tpr_RF_test = get_auc_scores(df_test.Exited, RF.predict(df_test.loc[:,
df_test.columns != 'Exited']),
                                                       RF.predict_proba(df_test.loc[:, df_test.columns !=
'Exited'])[:,1])
plt.figure(figsize = (12,6), linewidth= 1)
plt.plot(fpr_RF_test, tpr_RF_test, label = 'RF score: ' + str(round(auc_RF_test, 5)))
plt.plot([0,1], [0,1], 'k--', label = 'Random: 0.5')
```

```
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC Curve')
plt.legend(loc='best')
#plt.savefig('roc_results_ratios.png')
plt.show()


# конец
```