Рубежный контроль №2

Рысьева Елизавета Антоновна ИУ5-61Б

Вариант 15

In [1]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from sklearn.metrics import mean_absolute_error, mean_squared_error, median_
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.impute import SimpleImputer, MissingIndicator
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler,
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
```

In [9]:
```
df = pd.read_csv('bank_dataset.csv')
```

In [10]:
```
df.head()
```

Out[10]:

	userid	score	City	Gender	Age	Objects	Balance	Products	CreditCard	Lo
0	15677338	619	Ярославль	Ж	42	2	NaN	1	1	
1	15690047	608	Рыбинск	Ж	41	1	83807.86	1	0	
2	15662040	502	Ярославль	Ж	42	8	159660.80	3	1	
3	15744090	699	Ярославль	Ж	39	1	NaN	2	0	
4	15780624	850	Рыбинск	Ж	43	2	125510.82	1	1	

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   userid            10000 non-null  int64  
 1   score             10000 non-null  int64  
 2   City              10000 non-null  object 
 3   Gender            10000 non-null  object 
 4   Age               10000 non-null  int64  
 5   Objects           10000 non-null  int64  
 6   Balance           6383 non-null   float64
 7   Products          10000 non-null  int64  
 8   CreditCard        10000 non-null  int64  
 9   Loyalty           10000 non-null  int64  
 10  estimated_salary  10000 non-null  float64
 11  Churn             10000 non-null  int64  
dtypes: float64(2), int64(8), object(2)
memory usage: 937.6+ KB
```

In [13]:
```
#Кодирование категориальных признаков

df["City"] = df["City"].astype('category')

df["Gender"] = df["Gender"].astype('category')


#Назначить закодированную переменную новосу столбцу с помощью метода доступа
df["City_cat"] = df["City"].cat.codes
df["Gender_cat"] = df["Gender"].cat.codes
```

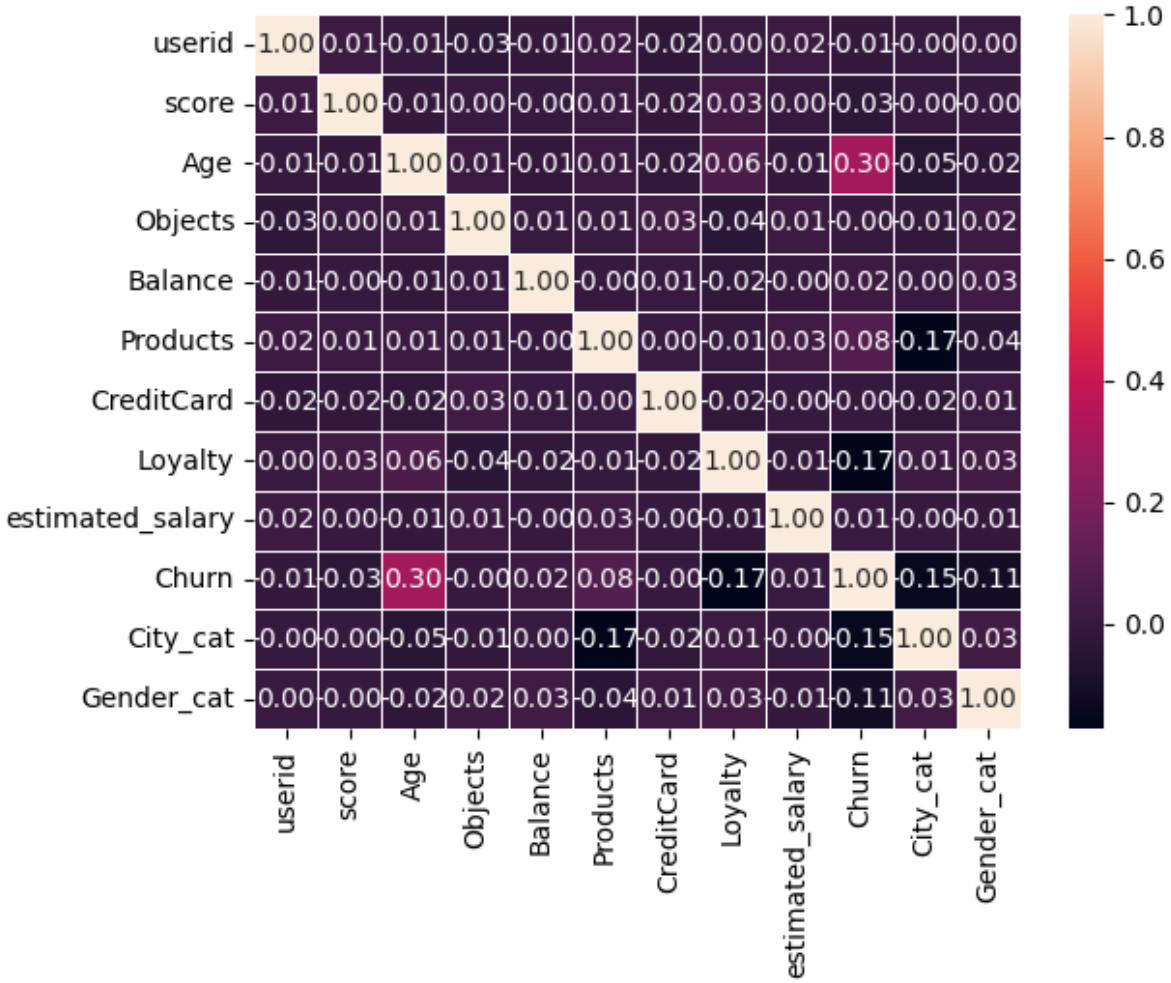
In [12]:
```
df.isnull().sum()
```

Out[12]:
```
userid                0
score                 0
City                  0
Gender                0
Age                   0
Objects               0
Balance            3617
Products              0
CreditCard            0
Loyalty               0
estimated_salary      0
Churn                 0
dtype: int64
```

In [14]:
```
df = df.drop(['City', 'Gender'], axis=1)
```

In [15]:
```
df = df.dropna()
```

In [16]:
```
corr = df.corr()
sns.heatmap(corr, linewidths=.5, annot=True, fmt=".2f")
plt.show()
```



1) С целевым признаком "Churn" наиболее коррелируют признаки "age". При построении модели машинного обучения перечисленные признаки будут наиболее информативными.

Разделение данных

Разделим данные на целевой столбец и признаки При построении предсказательных моделей исходные данные обычно разбиваются на обучающую ("training set") и контрольную ("test set") выборки. **Обучающая выборка** используется для построения математических отношений между некоторой переменной-откликом и предикторами, тогда как **контрольная (= "проверочная")** выборка служит для получения оценки прогнозных свойств модели на новых данных, т.е. данных, которые не были использованы для обучения модели.

In [17]:
```
X = df.drop(['Churn'], axis=1) #Наименования признаков
y = df['Churn'] # Значения признаков
```

In [18]:
```
# кодируем категориальные данные из строк в числа
le = LabelEncoder()
y = le.fit_transform(y)
```

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.20, sh
```

In [20]:
```
# Размер обучающей выборки
X_train.shape, y_train.shape
```

Out[20]:
```
((5106, 11), (5106,))
```

In [21]:
```
# Размер тестовой выборки
X_test.shape, y_test.shape
```

Out[21]:
```
((1277, 11), (1277,))
```

Обучите:

1) одну из линейных моделей,
2) случайный лес

Оцените качество моделей с помощью трех подходящих для задачи метрик. Сравните качество полученных моделей.

In [22]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,accuracy_score
from sklearn.svm import SVC, NuSVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, expo
from sklearn.model_selection import GridSearchCV
from sklearn import tree
from IPython.core.display import HTML
from sklearn.tree import export_text
from operator import itemgetter
```

1) Линейная модель: Допустим, у нас есть задача регрессии, и мы хотим обучить линейную модель на данных. Мы можем использовать, например, Ridge регрессию. Одним из гиперпараметров этой модели является alpha - коэффициент регуляризации. Мы можем использовать GridSearchCV для подбора оптимального значения alpha с помощью кросс-валидации.

In [23]:
```
LinearRegression_model=LinearRegression()
LinearRegression_model.fit(X_train,y_train)
accuracy_LinearRegression=LinearRegression_model.score(X_test,y_test)
accuracy_LinearRegression
```

Out[23]:
```
0.1084185718701598
```

In [24]:
```
def test_model(model):
    print("mean_absolute_error:",
          mean_absolute_error(y_test, model.predict(X_test)))
    print("median_absolute_error:",
          median_absolute_error(y_test, model.predict(X_test)))
    print("r2_score:",
          r2_score(y_test, model.predict(X_test)))
```

In [26]:
```
test_model(LinearRegression_model.fit(X_train,y_train))
```

```
mean_absolute_error: 0.3203728237454674
median_absolute_error: 0.2601717473067357
r2_score: 0.1084185718701598
```

Средняя абсолютная ошибка (MAE) равна 0.32, что означает, что модель в среднем ошибается на 0.32 единицы при прогнозировании целевой переменной.

Медианная абсолютная ошибка (MedAE) равна 0.26, что означает, что половина ошибок модели меньше 0.26, а другая половина - больше 0.26. Коэффициент детерминации (R2) равен 0.11, что означает, что модель объясняет только 11% дисперсии целевой переменной. Это может быть не очень хорошим результатом, если требуется точное прогнозирование. Однако, если целью является просто получение общей тенденции, то такой результат может быть достаточным.

Произведите для каждой модели подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации.

In [30]:
```
model = LinearRegression()

param_grid = {'normalize': [True, False]}

grid_search = GridSearchCV(model, param_grid, cv=5)

grid_search.fit(X, y)

best_params = grid_search.best_params_

cv_score = cross_val_score(grid_search.best_estimator_, X, y, cv=5).mean()

y_pred = grid_search.best_estimator_.predict(X_test)

accuracy_LinearRegression = grid_search.best_estimator_.score(X_test,y_test)

print("Наилучшие параметры: {}".format(grid_search.best_params_))
print("Оценка точности на кросс-валидации: {:.2f}".format(grid_search.best_score_
print(accuracy_LinearRegression)
```

```
Наилучшие параметры: {'normalize': False}
Оценка точности на кросс-валидации: 0.15
0.1134862039776614
```

Сравните качество полученных моделей с качеством моделей

In [34]:
```
model = LinearRegression_model.fit(X_train,y_train)
predictions = model.predict(X_test)
print((np.sqrt(mean_squared_error(y_test, predictions))))
```

```
0.40031099500911765
```

In [35]:
```
model=LinearRegression(normalize = False)
```

In [36]:
```
model.fit(X_train, y_train)
predictions = model.predict(X_test)
print(np.sqrt(mean_squared_error(y_test, predictions)))
```

```
0.40031099500911765
```

Сравнение выводов:

- Оба вывода получили одинаковую точность

Виды ансамблевых методов

1) Бэггинг. В этом случае однородные модели обучают на разных наборах данных и объединяют. Получают прогноз путём усреднения. Если использовать в качестве слабого ученика деревья решений, то получится случайный лес RandomForestClassifier / RandomForestRegressor.

In [37]:
```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassif
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_predict
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
```

Модель бэггинга - случайный лес (Random Forest):

In [38]:
```
# Создаем модель случайного леса с 100 деревьями
rf_model = RandomForestClassifier(n_estimators=100)

# Обучаем модель на тренировочных данных
rf_model.fit(X_train, y_train)

# Оцениваем качество модели на тестовых данных
accuracy = rf_model.score(X_test, y_test)
print('Accuracy: {:.2f}%'.format(accuracy*100))
```

```
Accuracy: 83.56%
```

Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

In [40]:
```
model = RandomForestClassifier()

param_grid = {
    'n_estimators': [200, 700],
    'max_features': ['auto', 'sqrt', 'log2']
}

grid_search = GridSearchCV(model, param_grid=param_grid, cv=5)

grid_search.fit(X_train, y_train)

accuracy_RandomForestClassifier = grid_search.best_estimator_.score(X_test,y

print("Наилучшие параметры: {} ", grid_search.best_params_)
print("Оценка точности на кросс-валидации: {:.2f}".format(grid_search.best_score_
```

```
Наилучшие параметры: {}  {'max_features': 'auto', 'n_estimators': 200}
Оценка точности на кросс-валидации: 0.84
```

Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

In [41]:
```
models = [
           ['RandomForestClassifier :',RandomForestClassifier()]
         ]
```

In [42]:
```
for name,model in models:
    model = model
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(name, (np.sqrt(mean_squared_error(y_test, predictions))))
```

```
RandomForestClassifier : 0.4055217932784942
```

In [43]:
```
models = [['RandomForestClassifier :',RandomForestClassifier(max_features =
         ]
```

In [44]:
```
for name,model in models:
    model = model
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(name, (np.sqrt(mean_squared_error(y_test, predictions))))
```

```
RandomForestClassifier : 0.40261478953820645
```