Федеральное государственное автономное

образовательное учреждение высшего образования

«Пермский государственный национальный исследовательский университет» (ПГНИУ)

Региональный институт непрерывного образования (РИНО ПГНИУ)

Цифровая кафедра

Выпускная аттестационная (квалификационная) работа

по курсу профессиональной переподготовки «Анализ данных»

**АНАЛИЗ ВЛИЯНИЯ ОНЛАЙН-ИГР НА ТРЕВОЖНЫЕ РАССТРОЙСТВА**

Разработчики проекта:

Шилова Лада Сергеевна

Пермь, 2023

**Оглавление**

[ПАСПОРТ ПРОЕКТА 3](#_Toc186058186)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_Toc186058187)

[Анализ проблемы исследования 4](#_Toc186058188)

[Исходные данные 5](#_Toc186058189)

[Реализация проекта 6](#_Toc186058190)

[Этап 1. Предобработка данных перед анализом 6](#_Toc186058191)

[Этап 2. Анализ данных 8](#_Toc186058192)

[Этап 3. Построение моделей 13](#_Toc186058193)

[Заключение 17](#_Toc186058194)

[Список использованных источников и литературы 19](#_Toc186058195)

[Приложения 20](#_Toc186058196)

**ПАСПОРТ ПРОЕКТА**

**Название проекта:**Анализ влияния онлайн-игр на тревожные расстройства

**Сведения об авторах:**Шилова Лада Сергеевна

**Цель:**Проанализировать данные о распространенности тревожных расстройств среди игроков в онлайн игры. Построить регрессионную модель, позволяющую с высокой точностью прогнозировать значение диагностического критерия генерализованного тревожного расстройства.

**Задачи:**

1. Проанализировать предметную область, пояснить актуальность.

2. Ознакомиться с датасетом и осуществить предобработку данных.

3. Проанализировать данные. Провести корреляционный анализ, визуализировать данные, устранить выбросы

4. Построить регрессионные модели. Оценить результаты. Выбрать наилучшую модель и оценить ее качество.

5. Сформулировать выводы

**Краткое описание проекта:**

Необходимо проанализировать данные о распространенности тревожных расстройств среди игроков в онлайн игры. Определить, возможно ли описать зависимость значения диагностического критерия GAD\_T от игровых привычек респондента, его социо-демографических характеристик, а также его балла удовлетворенности жизнью и диагностических критериев социофобии (SPIN). Интерпретировать полученные результаты. Сделать выводы.

**Конкретные ожидаемые результаты:**

Регрессионная модель, предсказывающая балл GAD\_T.

**СОДЕРЖАНИЕ ПРОЕКТА**

**Анализ проблемы исследования**

В современном мире онлайн-игры занимают значимое место в жизни многих людей. Именно поэтому так важно понимать, как это занятие влияет на психическое здоровье человека. Связь между играми (в том числе и онлайн-играми) и уровнем тревожности является широко обсуждаемой темой, которая вызывает особый интерес у общества. Именно в этом и заключается актуальность данной темы.

*Гипотеза:* Игровые привычки влияют на уровень тревожности онлайн-геймеров. Предполагается, что то, сколько времени человек тратит на онлайн-игры, играет он один или с друзьями, в шутеры или симулятор фермы, с целью расслабиться или ради заработка – всё это влияет на его уровень тревожности.

*Цель:* проанализировать связь между различными привычками оналйн-игроков, а также их социо-демографическими характеристиками и уровнем тревожности; построить модель, способную прогнозировать уровень тревожности на основе вышеупомянутых факторов.

*Задачи:*

1. Ознакомиться с датасетом и осуществить предварительную обработку данных.

2. Проанализировать данные для выявления закономерностей и взаимосвязей в данных. Провести корреляционный анализ, визуализировать данные, устранить выбросы.

3. Построить различные регрессионные модели. Оценить результаты и выбрать наилучшую модель. Оценить ее качество.

4. На основе полученных результатов сформулировать выводы.

**Исходные данные**

Используемый датасет, является результатом соцопроса более 13 тысяч респондентов, направленного на изучение взаимосвязи между поведением онлайн-геймеров и их психологическим состоянием.

Датасет включает в себя больше 30 параметров, которые можно условно поделить на 3 группы:

1. Социально-демографические характеристики респондентов

Данная группа включает в себя такие признаки, как возраст, пол, место жительства, уровень образования и занятость респондентов.

1. Их игровые привычки

Содержит данные о таких вещах, как: любимые игры, предпочитаемая игровая платформа, количество часов, затраченных на игры и т.п.

1. Критерии психологического здоровья и благополучия

Содержит ответы на отдельные вопросы и суммарные итоговые баллы по опросникам: о генерализованной тревожности, о социофобии, о нарциссизме, о удовлетворенности жизнью.

**Реализация проекта**

**Этап 1. Предобработка данных перед анализом**

Первым делом были подлкючены необходимые библиотеки. Далее, датасет был загружен из csv-файла с помощью pandas. Затем были выведены первые 5 строк и размер датасета, а также общая информация о типах данных.

df.head(5)

df.info()

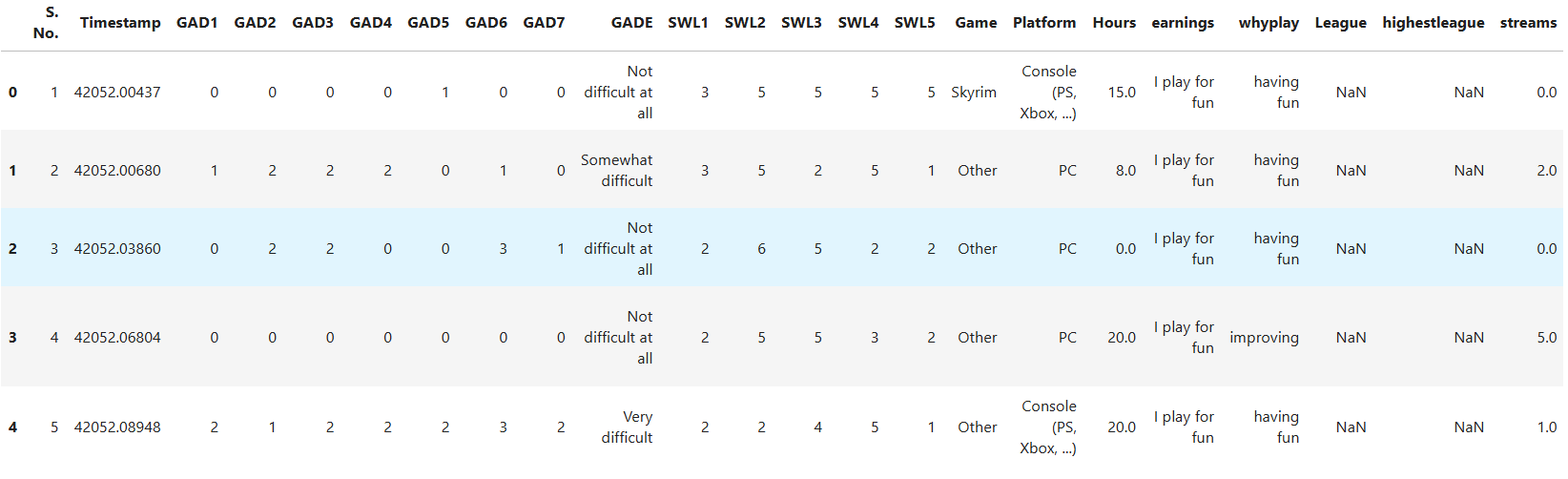


Рис.1 – Первоначальный датасет

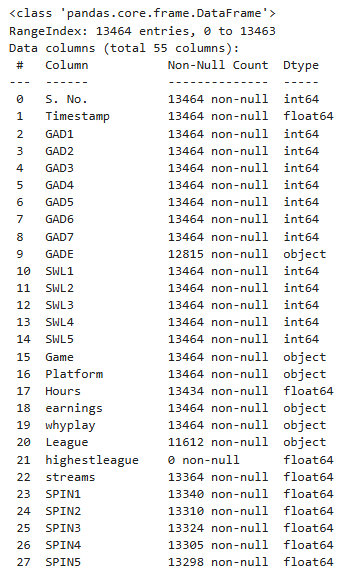


Рис.2 – Вывод df.info()

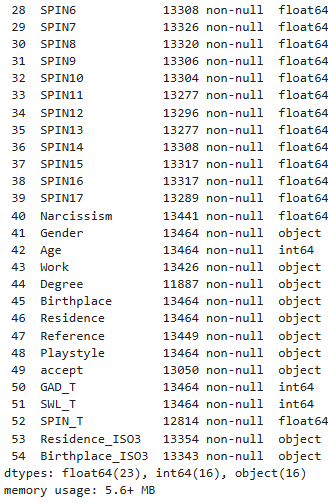


Рис.3 – Вывод df.info() (продолжение)

Затем были отброшены столбцы, не несущие значимой информации для дальнейшего анализа.

columns\_to\_drop=[

'Residence\_ISO3', # дублирует признак Residence

'Birthplace\_ISO3', # дублирует признак Birthplace

'accept', # согласие на обработку данных...

'highestleague', # все значения null

'Reference', # откуда респондент узнал про опрос

'Timestamp', # время

'S. No.', # порядковый номер респондента

'League' # лиги в каждой игре отличаются, так что особого смысла не несет + много пропущенных

]

df.drop(columns\_to\_drop, axis=1, inplace=True)

Далее были устранены пропущенные значения: пропущенные числовые значения заменены средним арифметическим, пропущенные категориальные – модой.

cols = cols\_to\_check + ['Narcissism', 'Degree', 'Work', 'GADE'] # колонки, которые будут заполнены

for col in cols:

df[col] = df[col].fillna(df[col].mode()[0])

df.drop\_duplicates(inplace=True)

Были удалены дубликаты. Затем, были обработаны текстовые данные в столбцах earnings, whyplay и Playstyle. Редкие значения (ответы в свободной форме) были заменены на самые часто встречающиеся.

def **standardize\_text**(df, text\_field):

df[text\_field] = df[text\_field].str.replace(r"http\S+", "")

df[text\_field] = df[text\_field].str.replace(r"http", "")

df[text\_field] = df[text\_field].str.replace(r"@\S+", "")

df[text\_field] = df[text\_field].str.replace(r"[^A-Za-z0-9(),!?@\'\`\"\\_\n]", " ")

df[text\_field] = df[text\_field].str.replace(r"@", "at")

df[text\_field] = df[text\_field].str.lower()

return df

df = standardize\_text(df, 'earnings')

most\_popular = df['earnings'].value\_counts()[:5].index

df.loc[~df['earnings'].isin(most\_popular), 'earnings'] = most\_popular[0]

Далее категориальные признаки были переведены в числовые с помощью LabelEncoder

enc = LabelEncoder()

for col in categorical:

df[col] = enc.fit\_transform(df[col])

**Этап 2. Анализ данных**

Рассмотрим тепловые карты матриц корреляций. На первой тепловой карте, рассмотрим зависимость между баллами GAD\_T(общая тревожность), SPIN\_T(социальная тревожность), SWL\_T(удовлетворенность жизнью) и социо-демографическими факторами и игровыми привычками. (В качестве метода оценки корреляции используем коэффициент Спирмена.)

correlations = df[features].corr(method='spearman') sns.heatmap(correlations, annot=True, mask=mask)

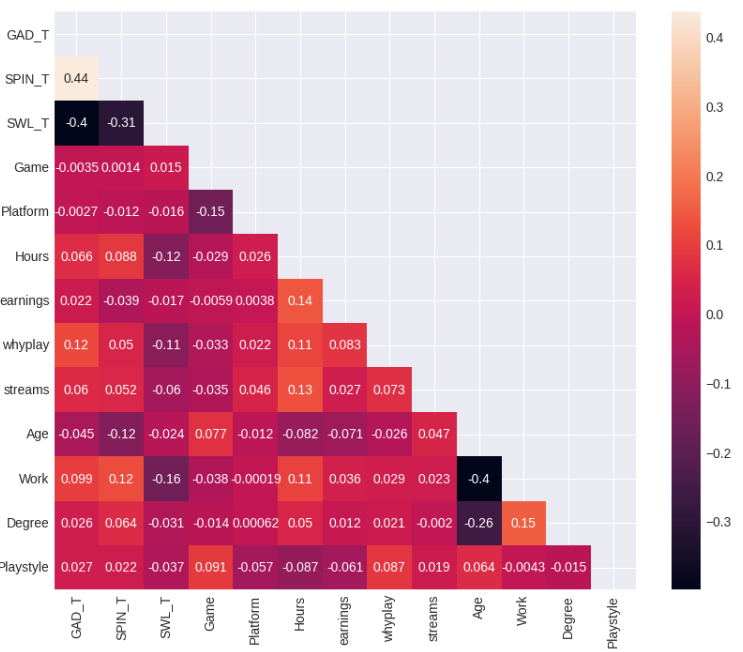


Рис 4. - Тепловая карта 1

Можно прийти к выводу, что социо-демографические факторы и игровые привычки почти никак не влияют на уровень общей тревожности среди онлайн-игроков. Ни один из признаков не имеет с признаком GAD\_T коэффициента корреляции Спирмена выше 0.12 по модулю.

У уровня социальной тревожности и уровня удовлетворенности жизнью также почти не прослеживается зависимости с привычками онлайн-игроков. Ни один из признаков не имеет с признаками SWL\_T и SPIN\_T коэффициента корреляции Спирмена выше 0.16 и 0.12 по модулю, соответсвенно.

Но при этом прослеживается слабая зависимость между самими баллами GAD\_T, SPIN\_T, SWL\_T. Рассмотрим зависимость между финальными баллами GAD\_T(общая тревожность), SPIN\_T(социальная тревожность) и ответами на опросник SWL о удовлетворенности жизнью.

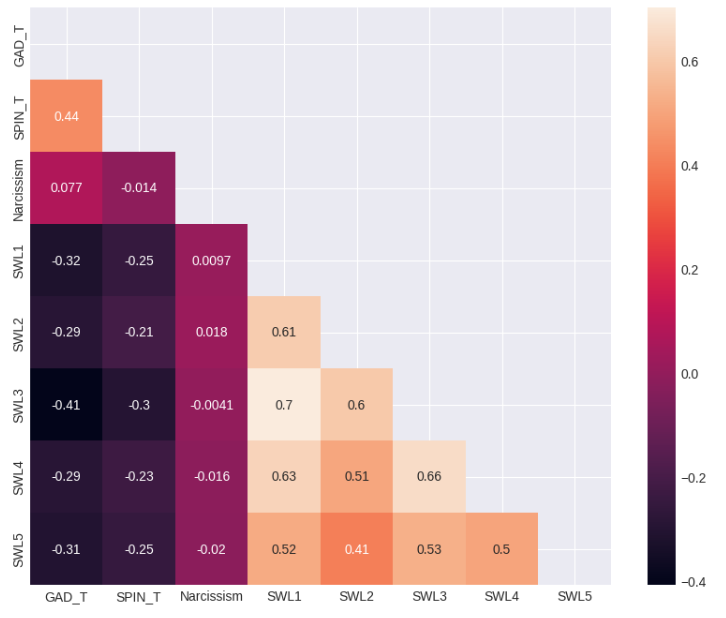


Рис 5. - Тепловая карта 2

Как и ожидалось, между ответами на опросник об удовлетворенности жизнью и баллами SWL\_T, GAD\_T прослеживается отрицательная корелляция, что логично - чем довольнее жизнью человек, тем меньше он тревожится. Менее интересно, но тоже стоит отметить, что ответы на опросник положительно коррелируют между собой.

Дальше рассмотрим распределения величин SWL\_T, GAD\_R, SPIN\_T.

fig, axes = plt.subplots(3, figsize=(6, 8))

sns.histplot(data=df, x="GAD\_T", ax=axes[0], color="indigo")

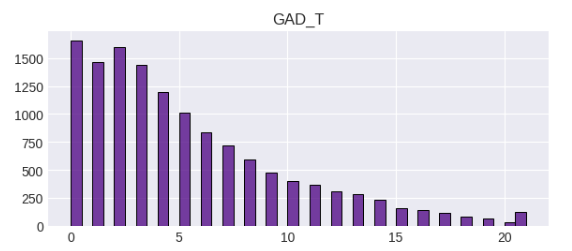


Рис. 6 – Гистограмма GAD\_R

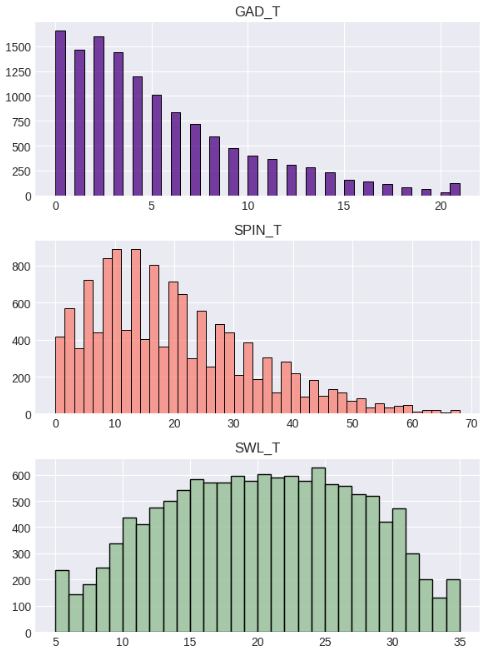


Рис. 7 – Гистограммы SWL\_T и SPIN\_T

Похоже, что GAD\_T имеет экспоненциальное распределение - большинству онлайн-геймеров свойственны низкие уровни генерализованного тревожного расстройства, а высокий уровень тревоги возникает реже.

Гистограмма SPIN\_T напоминает нормальное распределение с правосторонней асимметрией - большинство онлайн-геймеров имеет умеренно-низкие уровни социофобии, но есть небольшая группа с высокими уровнями.

Наконец, величина SWL\_T имеет нормальное распределение - большинство респондентов удовлетворены своей жизнью на среднем уровне.

Затем построим диаграммы типа «ящик с усами» (boxplot) для параметров Hours, streams и Age. Попытаемся обнаружить и устранить выбросы.

for i, col in enumerate(features):

sns.boxplot(ax=axes[i], y=df[col], color=colors[i])

plt.tight\_layout()

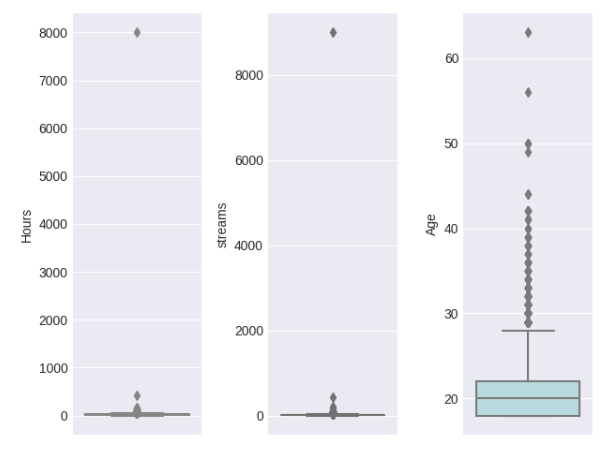


Рис. 8 – Диаграммы «ящик с усами» (Hours, streams и Age)

Видим, что выбросы есть. Удалим их методом межквартильного размаха.

def remove\_outliers(data, col):

lower\_q = data[col].quantile(0.25)

upper\_q = data[col].quantile(0.75)

IQR = upper\_q - lower\_q

lower\_bound = lower\_q - 1.5 \* IQR

upper\_bound = upper\_q + 1.5 \* IQR

data\_out = data.loc[(data[col] > lower\_bound) & (data[col] < upper\_bound)]

return data\_out

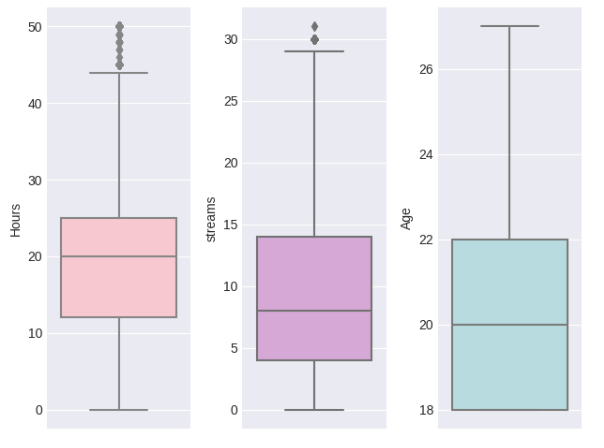
Посмотрим, как выглядят «ящики с усами» после удаления выбросов. 

Рис. 9 – Диаграммы «ящик с усами» (Hours, streams и Age) (2)

**Этап 3. Построение моделей**

Проведем нормализацию данных с помощью RobustScaler.

from sklearn.preprocessing import RobustScaler

list = ['Hours', 'Age', 'GAD\_T', 'SWL\_T', 'SPIN\_T']

sc = RobustScaler()

df[list] = sc.fit\_transform(df[list])

Выделим X и y. Разобьем данные на тренирововчную и тестовую выборки.

X = df.copy().drop(same, axis=1)

y = df['GAD\_T'].copy()

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=13)

Подготовим функцию для вывода метрик.

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

def metrics(y\_test, y\_pred, X\_test):

n = X\_test.shape[0]

p = X\_test.shape[1]

r2 = r2\_score(y\_test, y\_pred)

adjusted\_r2 = 1-(1-r2)\*(n-1)/(n-p-1)

print(f"MAE:\t{mean\_absolute\_error(y\_test, y\_pred):.2f}")

print(f"MSE:\t{mean\_squared\_error(y\_test, y\_pred):.2f}")

print(f"RMSE:\t{np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.2f}")

print(f"\nR2:\t{r2:.2f}")

print(f"adj R2:\t{adjusted\_r2:.2f}\n")

Рассмотрим следующие модели:

- линейная регрессия;

- полиномиальная регрессия;

- метод опорных векторов;

- ElasticNet;

- случайный лес;

- градиентный бустинг.

Для каждой модели: обучим модель на выборке train, получим предсказанные значения GAD\_T на выборке test, выведем метрики. Там, где требуется подбор гиперпараметров, будем использовать GridSearchCV.

# ЛИНЕЙНАЯ РЕГРЕССИЯ

lr = LinearRegression()

lr.fit(X\_train, y\_train)

y\_pred\_lr = lr.predict(X\_test)

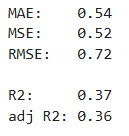


Рис. 10 – Метрики линейной регрессии

# ПОЛИНОМИАЛЬНАЯ РЕГРЕССИЯ

pf = PolynomialFeatures(degree=2)

polys = pf.fit\_transform(X)

X\_p\_train, X\_p\_test, y\_p\_train, y\_p\_test = train\_test\_split(polys, y, test\_size=0.3, random\_state=13)

lpr = LinearRegression()

lpr.fit(X\_p\_train, y\_p\_train)

y\_pred\_plr = lpr.predict(X\_p\_test)

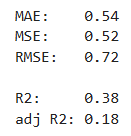


Рис. 11 – Метрики полиномиальной регрессии

# ELASTIC NET (Регуляризация L1 + L2)

en = ElasticNet(alpha=0.3)

en.fit(X\_train, y\_train)

y\_pred\_en = en.predict(X\_test)

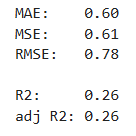


Рис. 12 – Метрики ElasticNet

# SVM (МЕТОД ОПОРНЫХ ВЕКТОРОВ)

sv = svm.SVR(kernel='rbf')

sv.fit(X\_train, y\_train)

y\_pred\_sv = sv.predict(X\_test)

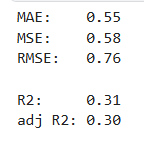


Рис. 13 – Метрики SVM

# Random Forest (СЛУЧАЙНЫЙ ЛЕС)

model = RandomForestRegressor()

params = {

'n\_estimators': [50, 100, 200],

'max\_depth': [None, 10, 20, 30],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(model, params, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

y\_pred\_rf = best\_model.predict(X\_test)

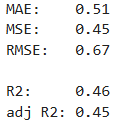


Рис. 14 – Метрики случайного леса

# ГРАДИЕНТНЫЙ БУСТИНГ

model = GradientBoostingRegressor(random\_state=42)

params = {

'n\_estimators': [50, 100, 150],

'learning\_rate': [0.01, 0.1, 0.2],

'max\_depth': [3, 4, 5]

}

grid\_search = GridSearchCV(model, params, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

grid\_search.fit(X\_train, y\_train)

best\_model\_gb = grid\_search.best\_estimator\_

y\_pred\_gb = best\_model\_gb.predict(X\_test)

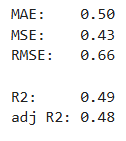


Рис. 14 – Метрики градиентного бустинга

В результате, самые относительно приличные метрики показали методы случайного леса и градинтного бустинга: R2 равное 0.46 и 0.49, соответственно. Значение R2 = 0.49 говорит о том, что модель объясняет 49% дисперсии предсказываемой величины.

**Заключение**

Основываясь на результатах проведенного регрессионного анализа, можно прийти к выводу, что выдвинутая гипотеза не оправдала себя. Как критерий уровня тревоги GAD\_T, так и критерий социофобии SPIN\_T и критерий удовлетворенности жизнью SWL\_T почти не зависят от игровых привычек онлайн-геймеров. Большинству игроков в онлайн игры свойственен низкий балл тревожности, в то время как баллы удовлетворенности жизнью среди респондентов имеют нормальное распределение.

Оказалось, что параметры GAD\_T, SPIN\_T и SWL\_T гораздо сильнее коррелируют друг с другом, чем с поведением игроков.

Таким образом, цель была достигнуто лишь частично. Был проведен анализ и было построено несколько регрессионных моделей, однако ни одной из них не удалось достигнуть высокой точности прогнозирования.

Поставленные задачи были решены.

**Список использованных источников и литературы**

* + - 1. Sofianti, Z. Online Gaming and Anxiety: A cross-cultural study / Z. Sofianti, M. H. Al Qadrie, L. Rahmawati // OSF Preprints. – URL: https://osf.io/vnbxk/ (дата обращения: 07.11.2024)
      2. Scikit-learn Developers. User Guide // Scikit-learn. – URL: https://scikit-learn.org/stable/user\_guide.html (дата обращения: 07.11.2024).
      3. Seaborn developers. Seaborn // Seaborn. – URL: https://seaborn.pydata.org/ (дата обращения: 07.11.2024)
      4. OTUS. Гиперпараметрический поиск и оптимизация моделей / OTUS // Хабр. – URL: https://habr.com/ru/companies/otus/articles/754402/ (дата обращения: 07.11.2024)
      5. Как работает самая популярная у дата саентистов модель машинного обучения: градиентный бустинг на пальцах // Хабр. – URL: https://habr.com/ru/articles/819869/ (дата обращения: 07.11.2024)

**Приложения**

**Приложение 1**

**Программный код**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

file = '/kaggle/input/online-gaming-anxiety-data/GamingStudy\_data.csv'

df = pd.read\_csv(file, encoding='ISO-8859-1')

df.head(5)

df.shape

df.info()

columns\_to\_drop = [

    'Residence\_ISO3',

    'Birthplace\_ISO3',

    'accept',

    'highestleague',

    'Reference',

    'Timestamp',

    'S. No.',

    'League'

]

df.drop(columns\_to\_drop, axis=1, inplace=True)

df.isnull().sum()

cols\_to\_check = [f'SPIN{i}' for i in range(1, 18)]

df.dropna(subset=cols\_to\_check, how='all', axis=0, inplace=True)

cols = cols\_to\_check + ['Narcissism', 'Degree', 'Work', 'GADE']

for col in cols:

    df[col] = df[col].fillna(df[col].mode()[0])

cols = ['streams', 'Hours']

for col in cols:

    df[col] = df[col].fillna(df[col].mean())

df['SPIN\_T'] = df['SPIN\_T'].fillna(df[cols\_to\_check].sum(axis=1))

df.duplicated().sum()

df.drop\_duplicates(inplace=True)

def standardize\_text(df, text\_field):

    df[text\_field] = df[text\_field].str.replace(r"http\S+", "")

    df[text\_field] = df[text\_field].str.replace(r"http", "")

    df[text\_field] = df[text\_field].str.replace(r"@\S+", "")

    df[text\_field] = df[text\_field].str.replace(r"[^A-Za-z0-9(),!?@\'\`\"\\_\n]", " ")

    df[text\_field] = df[text\_field].str.replace(r"@", "at")

    df[text\_field] = df[text\_field].str.lower()

    return df

most\_popular = df['Playstyle'].value\_counts()[:9].index

df.loc[~df['Playstyle'].isin(most\_popular), 'Playstyle'] = most\_popular[0]

df = standardize\_text(df, 'earnings')

most\_popular = df['earnings'].value\_counts()[:5].index

df.loc[~df['earnings'].isin(most\_popular), 'earnings'] = most\_popular[0]

df = standardize\_text(df, 'whyplay')

most\_popular = df['whyplay'].value\_counts()[:8].index

df.loc[~df['whyplay'].isin(most\_popular), 'whyplay'] = most\_popular[0]

from sklearn.preprocessing import LabelEncoder

categorical = df.select\_dtypes(include='object').columns

enc = LabelEncoder()

for col in categorical:

    df[col] = enc.fit\_transform(df[col])

features = ['GAD\_T', 'SPIN\_T', 'SWL\_T', 'Game', 'Platform', 'Hours', 'earnings', 'whyplay', 'streams', 'Age', 'Work', 'Degree', 'Playstyle']

correlations = df[features].corr(method='spearman')

mask = np.triu(np.ones\_like(correlations))

plt.subplots(figsize=(10, 8))

sns.heatmap(correlations, annot=True, mask=mask)

features = ['GAD\_T', 'SPIN\_T', 'Narcissism'] + [f'SWL{i}' for i in range(1, 6)]

correlations = df[features].corr(method='spearman')

mask = np.triu(np.ones\_like(correlations))

plt.subplots(figsize=(10, 8))

sns.heatmap(correlations, annot=True, mask=mask)

fig, axes = plt.subplots(3, figsize=(6, 8))

sns.histplot(data=df, x="GAD\_T", ax=axes[0], color="indigo")

axes[0].set\_title('GAD\_T')

sns.histplot(data=df, x="SPIN\_T", ax=axes[1], color="salmon")

axes[1].set\_title('SPIN\_T')

sns.histplot(data=df, x="SWL\_T", ax=axes[2], color="darkseagreen")

axes[2].set\_title('SWL\_T')

plt.tight\_layout()

features = ['Hours', 'streams', 'Age']

colors = ['pink', 'plum', 'powderblue']

fig, axes = plt.subplots(1, 3)

axes = axes.flatten()

for i, col in enumerate(features):

    sns.boxplot(ax=axes[i], y=df[col], color=colors[i])

plt.tight\_layout()

def remove\_outliers(data, col):

    lower\_q = data[col].quantile(0.25)

    upper\_q = data[col].quantile(0.75)

    IQR = upper\_q - lower\_q

    lower\_bound = lower\_q - 1.5 \* IQR

    upper\_bound = upper\_q + 1.5 \* IQR

    data\_out = data.loc[(data[col] > lower\_bound) & (data[col] < upper\_bound)]

    return data\_out

for col in features:

    df = remove\_outliers(df, col)

fig, axes = plt.subplots(1, 3)

axes = axes.flatten()

for i, col in enumerate(features):

    sns.boxplot(ax=axes[i], y=df[col], color=colors[i])

plt.tight\_layout()

colors = ['palegreen', 'moccasin', 'lightcoral']

fig, axes = plt.subplots(3)

for i, col in enumerate(features):

    sns.histplot(ax=axes[i], x=df[col], color=colors[i])

    axes[0].set\_title(col)

plt.tight\_layout()

from sklearn.preprocessing import RobustScaler

list = ['Hours', 'Age', 'GAD\_T', 'SWL\_T', 'SPIN\_T']

sc = RobustScaler()

df[list] = sc.fit\_transform(df[list])

same = ['GAD1', 'GAD2', 'GAD3', 'GAD4', 'GAD5', 'GAD6', 'GAD7', 'GAD\_T']

X = df.copy().drop(same, axis=1)

y = df['GAD\_T'].copy()

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=13)

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

def metrics(y\_test, y\_pred, X\_test):

    n = X\_test.shape[0]

    p = X\_test.shape[1]

    r2 = r2\_score(y\_test, y\_pred)

    adjusted\_r2 = 1 - (1 - r2) \* (n - 1) / (n - p - 1)

    print(f"MAE:\t{mean\_absolute\_error(y\_test, y\_pred):.2f}")

    print(f"MSE:\t{mean\_squared\_error(y\_test, y\_pred):.2f}")

    print(f"RMSE:\t{np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.2f}")

    print(f"\nR2:\t{r2:.2f}")

    print(f"adj R2:\t{adjusted\_r2:.2f}\n")

from sklearn.linear\_model import LinearRegression

lr = LinearRegression()

lr.fit(X\_train, y\_train)

y\_pred\_lr = lr.predict(X\_test)

metrics(y\_test, y\_pred\_lr, X\_test)

from sklearn.preprocessing import PolynomialFeatures

pf = PolynomialFeatures(degree=2)

polys = pf.fit\_transform(X)

X\_p\_train, X\_p\_test, y\_p\_train, y\_p\_test = train\_test\_split(polys, y, test\_size=0.3, random\_state=13)

lpr = LinearRegression()

lpr.fit(X\_p\_train, y\_p\_train)

y\_pred\_plr = lpr.predict(X\_p\_test)

metrics(y\_p\_test, y\_pred\_plr, X\_p\_test)

from sklearn.linear\_model import ElasticNet

en = ElasticNet(alpha=0.3)

en.fit(X\_train, y\_train)

y\_pred\_en = en.predict(X\_test)

metrics(y\_test, y\_pred\_en, X\_test)

sv = svm.SVR(kernel='rbf')

sv.fit(X\_train, y\_train)

y\_pred\_sv = sv.predict(X\_test)

metrics(y\_test, y\_pred\_sv, X\_test)

from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from sklearn.model\_selection import GridSearchCV

model = RandomForestRegressor()

params = {

    'n\_estimators': [50, 100, 200],

    'max\_depth': [None, 10, 20, 30],

    'min\_samples\_split': [2, 5, 10],

    'min\_samples\_leaf': [1, 2, 4]

}

grid\_search = GridSearchCV(model, params, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

y\_pred\_rf = best\_model.predict(X\_test)

metrics(y\_test, y\_pred\_rf, X\_test)

model = GradientBoostingRegressor(random\_state=42)

params = {

    'n\_estimators': [50, 100, 150],

    'learning\_rate': [0.01, 0.1, 0.2],

    'max\_depth': [3, 4, 5]

}

grid\_search = GridSearchCV(model, params, cv=5, n\_jobs=-1, scoring='neg\_mean\_squared\_error')

grid\_search.fit(X\_train, y\_train)

best\_model\_gb = grid\_search.best\_estimator\_

y\_pred\_gb = best\_model\_gb.predict(X\_test)

metrics(y\_test, y\_pred\_gb, X\_test)