ИСПОЛЬЗОВАНИЕ АЛГОРИТМА XGBOOST ДЛЯ ПРЕДСКАЗАНИЯ ЗАВЕРШЕНИЯ КУРСА ОБУЧАЮЩИМСЯ

USING THE XGBOOST ALGORITHM TO PREDICT THE COURSE COMPLETION OF A STUDENT

УДК 004.5

Макаров Д.А., студент, makdenis.1997@gmail.com, Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана, кафедра «Системы обработки информации и управления»

Makarov D.A., student, makdenis.1997@gmail.com, Russia, 105005, Moscow, MSTU N.E. Bauman, Department of Information Processing and Management Systems

Аннотация

В настоящее время большой популярностью пользуются различные онлайн-курсы. Однако, распространена ситуация, когда пользователь бросает прохождение курса и не получает сертификат. В данной статье будет использован датасет, основанный на реальных данных одной из платформ онлайн курсов. Будет произведена очистка данных и их визуализация, а также анализ. С использованием библиотеки XGBoost будет произведена попытка создания модели, которая будет предсказывать, завершит ли пользователь более 50% курса или нет.

Summary

Various online courses are very popular nowadays. However, it is common for a user to drop out of a course and not receive a certificate. In this article, a dataset

will be used based on real data from one of the platforms of online courses. The data will be cleaned and visualized, as well as analyzed. Using the XGBoost library, an attempt will be made to create a model that will predict whether the user will complete more than 50% of the course or not.

Ключевые слова: машинное обучение, датасет, очистка данных, XGBoost, дерево решений.

Key words: machine learning, dataset, data cleaning, XGBoost, decision tree.

Исследование датасета

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

Импорт данных из csv файла

```
df_course= pd.read_csv('/Users/denis/Downloads/courses.csv')
```

Просмотр типов данных. Данные представлены типами object, int, float.

```
df_course.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 19 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Launch Date                                   205 non-null    object
 1   Course title                                  205 non-null    object
 2   Teachers                                      205 non-null    object
 3   Course subject                                205 non-null    object
 4   Participants                                  205 non-null    int64
 5   50% course content accessed (audited)         205 non-null    int64
 6   Certified                                     205 non-null    int64
 7   % Audited                                     205 non-null    float64
 8   % Certified                                   205 non-null    float64
 9   % Certified of > 50% course content accessed  205 non-null    float64
 10  % Played video                                205 non-null    float64
 11  % Posted in forum                             205 non-null    float64
 12  % Grade higher than 0                         205 non-null    float64
 13  Total course hours                            205 non-null    float64
 14  Median hours for certification                205 non-null    float64
 15  Median age                                    205 non-null    float64
 16  % Male                                        205 non-null    float64
 17  % Female                                      205 non-null    float64
 18  % Bachelor's degree or higher                 205 non-null    float64
dtypes: float64(12), int64(3), object(4)
memory usage: 30.6+ KB
```

Рисунок 1. Типы данных

Просмотр значений NULL в данных, установленных с помощью тепловой карты:

```
plt.figure(figsize=(15,10))
sns.heatmap(df_course.isnull(),cmap="YlGnBu") [1]
```

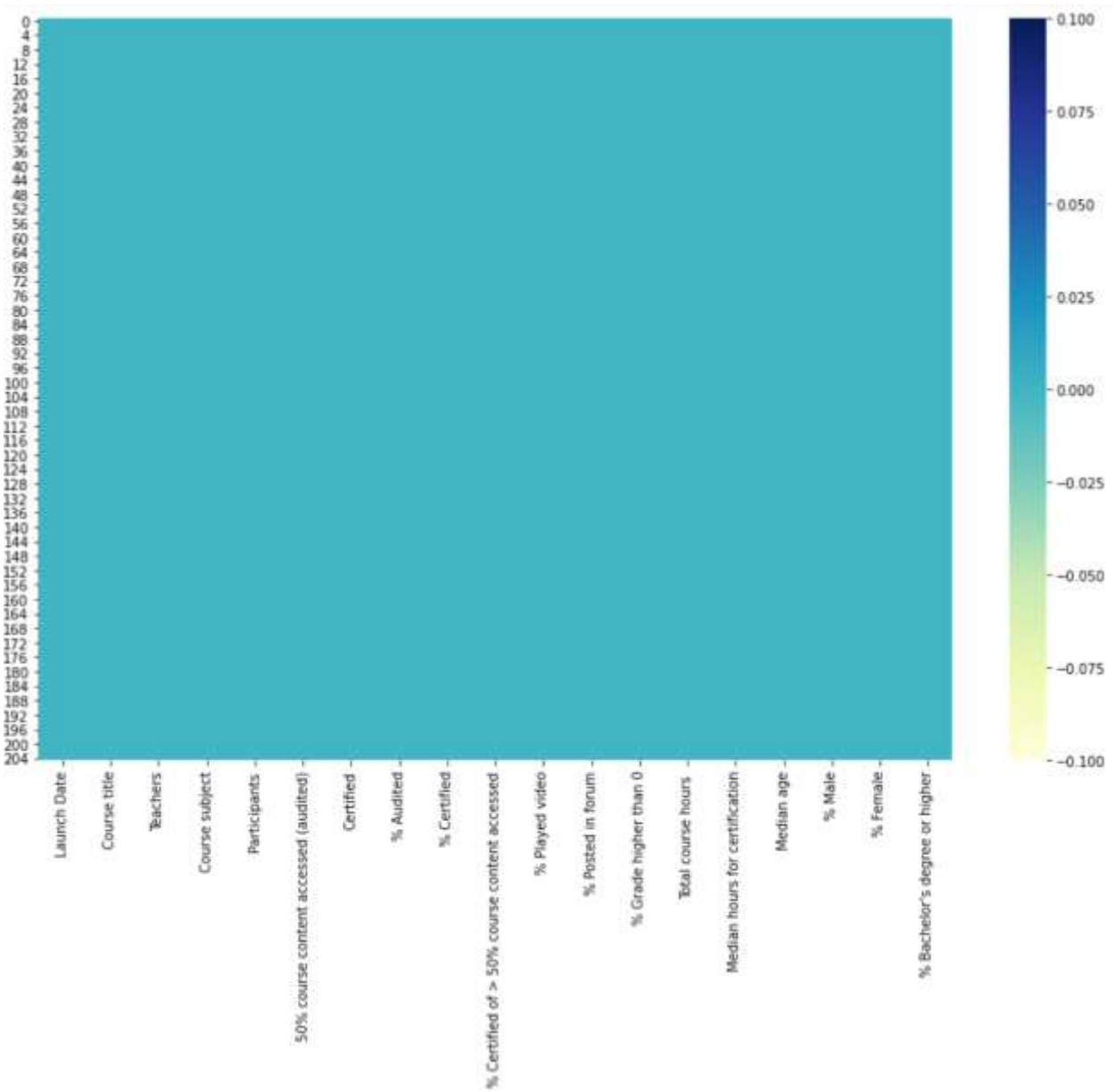

Рисунок 2. Поиск пустых значений

Убедились, что пустых ячеек нет.

Визуализация данных

Представим названия курсов в виде облака слов, где более часто используемые слова имеют больший размер, чем другие.

Отобразим категории курсов в виде облака слов.

```
from wordcloud import WordCloud, STOPWORDS


wordcloud = WordCloud(
                    stopwords =STOPWORDS,
                    background_color='white',
                    width=1200,
                    height=1000
                   ).generate(" ".join(df_course['Course subject']))

plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```



Рисунок 3. Облако слов

Судя по облакам слов, в названии курса чаще всего встречается слово «разработчик», а самой популярной категорией является бэкенд разработка и объектно-ориентированное программирование.

Построим матрицу корреляций между различными признаками.

df_course=df_course.drop(['% Certified','Course title','% Grade higher than 0'],axis=1) [2]

figure= plt.figure(figsize=(10,10))
sns.heatmap(df_course.corr(), annot=True,cmap="YlGnBu")

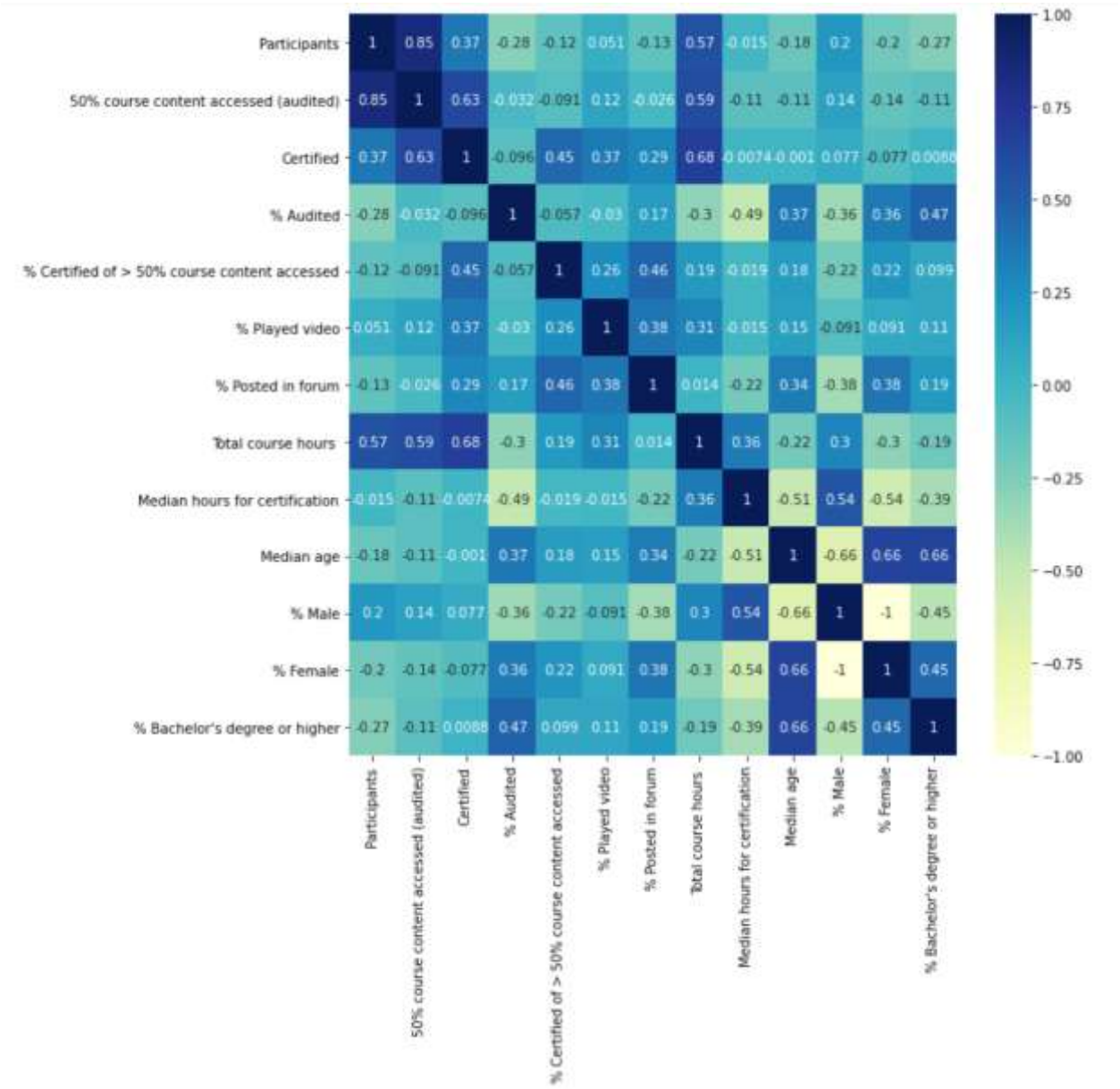

Рисунок 4. Матрица корреляций

Как видно из матрицы, сильной корреляции между признаками нет, будем продолжать исследование дальше.

Построим диаграммы размаха («ящик с усами») для признаков Course subject, Certified of > 50% course content accessed и Participants, Course subject.

```
figure= plt.figure(figsize=(20,10))
sns.boxenplot(x='Course subject',y='% Certified of > 50% course content accessed,data=df_course,palette="Blues")
```

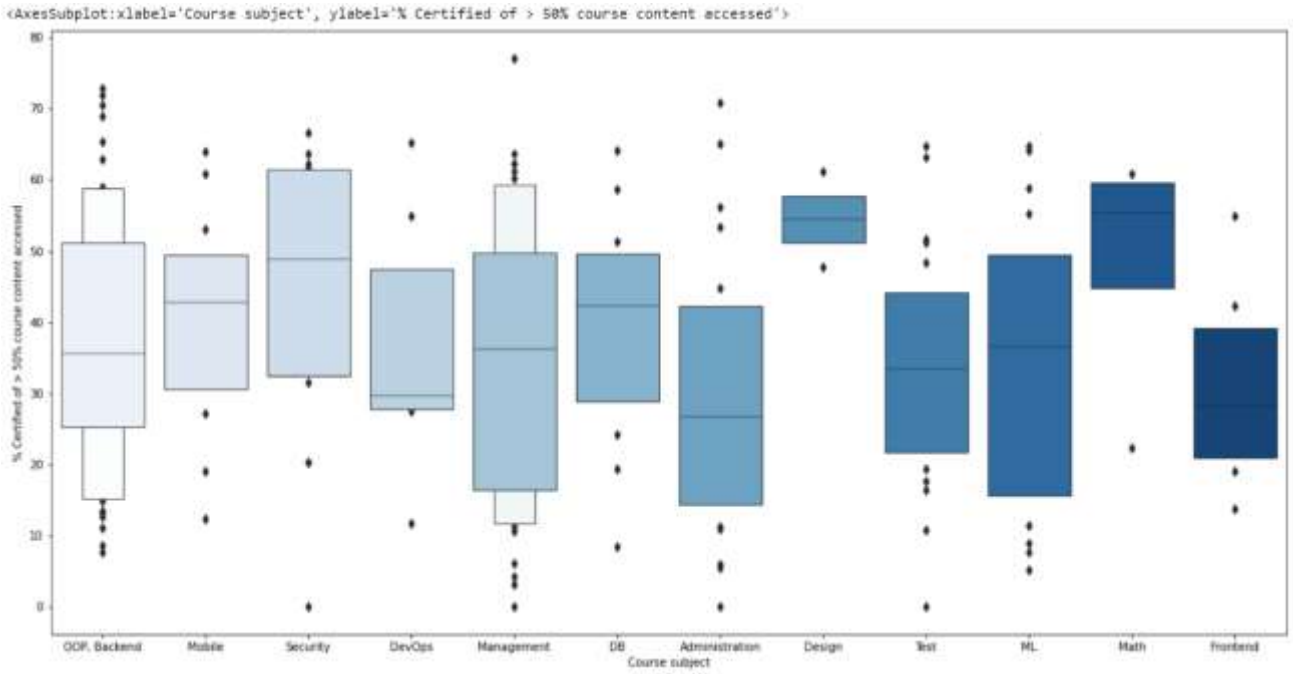

Рисунок 5. График "ящик с усами"

Данные графики в удобной форме показывают медиану (или, если нужно, среднее), нижний и верхний квартили, минимальное и максимальное значение выборки и выбросы. Расстояния между различными частями ящика позволяют определить степень разброса (дисперсии) и асимметрии данных и выявить выбросы.

График типа pairplot показывает отношения между всеми парами переменных.

```
df_pairplot_cols=df_course[['Course subject','50% course content accessed (audited)','% Certified of > 50% course content accessed','% Female','% Male','Median age']] [3]
```

```
plt.figure(figsize=(20,20))
sns.pairplot(df_pairplot_cols,hue='Course subject',palette='rainbow')
```

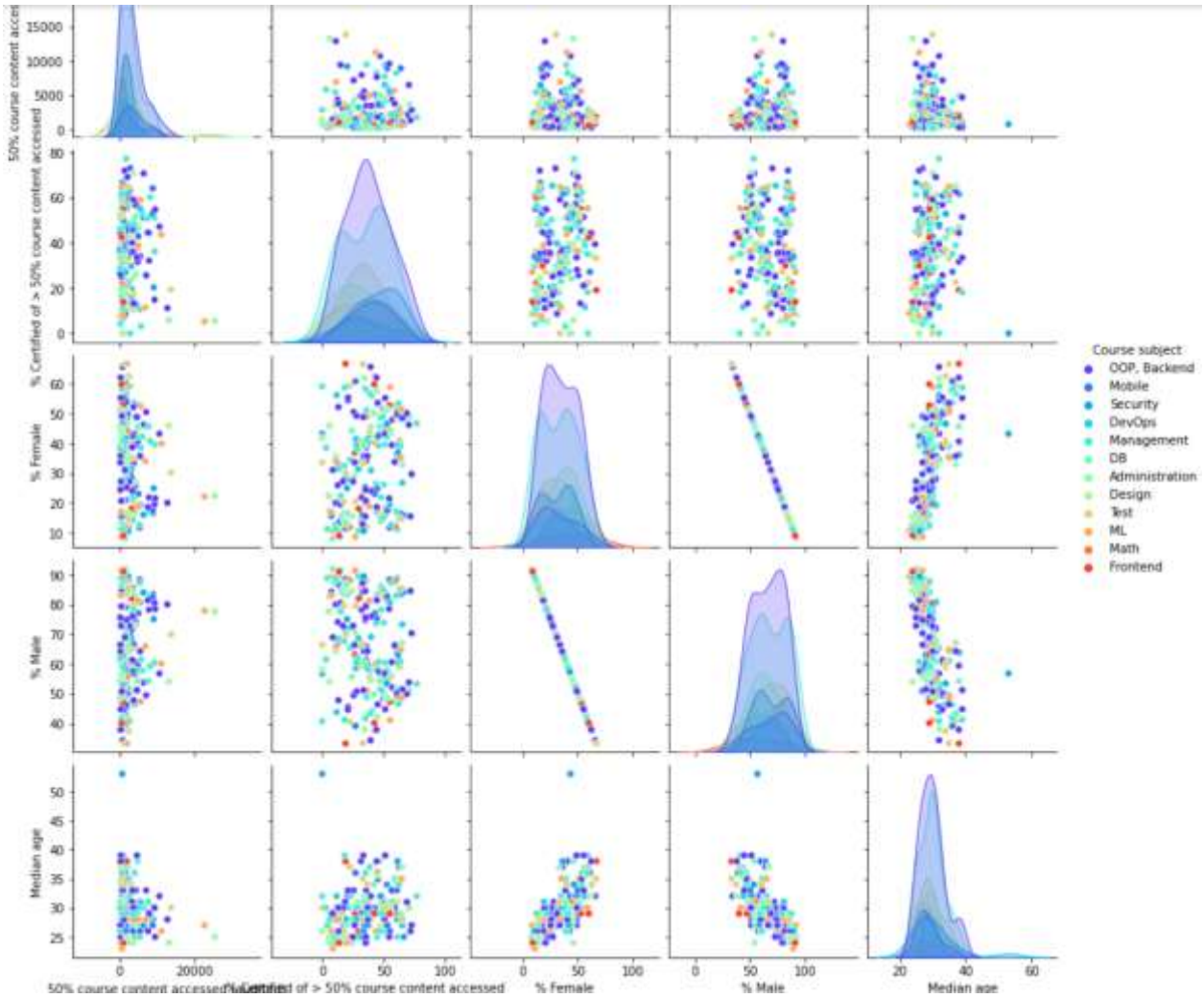

Рисунок 6. Отношение между всеми парами переменных

Построим kdeplot для Median hours for certification и % Certified of > 50% course content accessed.

График оценки плотности ядра (KDE) - это метод визуализации распределения наблюдений в наборе данных, аналогичный гистограмме. KDE представляет данные с помощью непрерывной кривой плотности вероятности в одном или нескольких измерениях.

По сравнению с гистограммой KDE может создавать график, который менее загроможден и более понятен, особенно при рисовании нескольких распределений. Но он может внести искажения, если основное распределение ограничено или негладко. Как и в случае гистограммы, качество представления также зависит от выбора хороших параметров сглаживания.

Построим kdeplot для признаков «Median age» и «% Certified of > 50% course content accessed».

```
x= df_course['Median age']
y= df_course['% Certified of > 50% course content accessed']
cmap = sns.cubehelix_palette(light=1, as_cmap=True)
plt.figure(figsize=(10,10))
sns.kdeplot(x, y, cmap=cmap, shade=True);
```

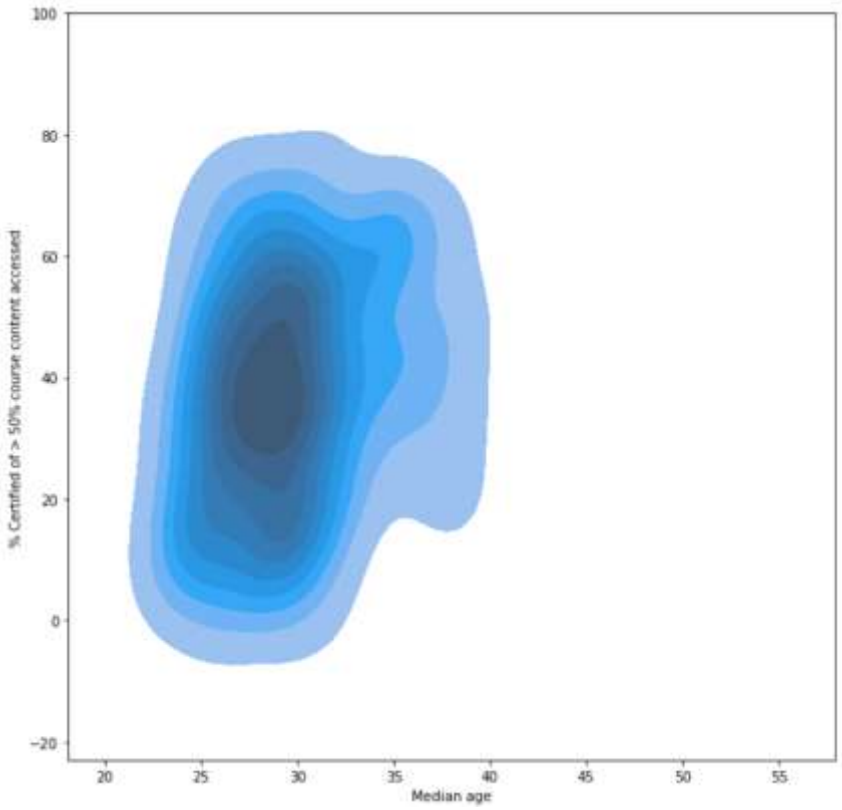

Рисунок 7. Kdeplot

По графику видно, что средний возраст составляет около 27 лет.

Подготовка данных для машинного обучения

Произведем удаление ненужных столбцов (Certified','50% course content accessed (audited)','Teachers','Launch Date','% Played video')

```
df_XGB = df_course.drop(['Certified','50% course content accessed (audited)','Teachers','Launch Date','% Played video'],axis=1)
```

Произведем преобразование категориальных переменных в серии нулей и единиц, что значительно упрощает их количественное определение и сравнение.

```
CourseSubject = pd.get_dummies(df_XGB['Course subject'],drop_first=True)
df_XGB.drop(['Course subject'],axis=1,inplace=True)
df_XGB = pd.concat([df_XGB,CourseSubject],axis=1)
```

Библиотека XGBoost

XGBoost - это оптимизированная распределенная библиотека повышения градиента, разработанная для обеспечения высокой эффективности, гибкости и портативности. Он реализует алгоритмы машинного обучения в рамках платформы Gradient Boosting. XGBoost обеспечивает усиление параллельного дерева (также известное как GBDT, GBM), которое позволяет быстро и точно решить многие проблемы data science. Один и тот же код работает в основной распределенной среде (Hadoop, SGE, MPI) и может решать проблемы, выходящие за рамки множества примеров. [4]

В основе XGBoost лежит алгоритм градиентного бустинга деревьев решений. Градиентный бустинг — это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. Обучение ансамбля проводится последовательно в отличие, например от бэггинга. На каждой итерации вычисляются отклонения предсказаний уже обученного ансамбля на обучающей выборке. Следующая модель, которая будет добавлена в ансамбль будет предсказывать эти отклонения. Таким образом, добавив

предсказания нового дерева к предсказаниям обученного ансамбля мы можем уменьшить среднее отклонение модели, которое является целью оптимизационной задачи. [5]

Разделим данные на обучающую и тестовую выборки.

```
from sklearn.model_selection import train_test_split
x= df_XGB
y=df_XGB['% Certified of > 50% course content accessed']
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.4, ran-
dom_state=109)

import xgboost as xgb
train= xgb.DMatrix(x_train,label=y_train)
test = xgb.DMatrix(x_test, label= y_test)
```

Список гиперпараметров XGBoost

learning_rate: уменьшение размера шага, используемое для предотвращения переобучения.

max_depth: определяет, насколько глубоко каждое дерево может расти во время любого раунда повышения.

subsample: процент использованных образцов на дерево. Низкое значение может привести к неполному оснащению.

colsample_bytree: процент функций, используемых в дереве. Высокое значение может привести к переобучению.

n_estimators: количество деревьев.

objective: определяет функцию потерь, которая будет использоваться, например, линейная для задач регрессии, логистическая для задач классификации с единственным решением, двоичная для задач классификации с вероятностью.

XGBoost также поддерживает параметры регуляризации, чтобы наказывать модели по мере их усложнения и сводить их к простым (экономным) моделям.

gamma: контролирует, будет ли данный узел разделен на основе ожидаемого сокращения потерь после разделения. Чем выше значение, тем меньше расщеплений.

alpha: L1 регуляризация весов листьев. Большое значение ведет к большей регуляризации.

lambda: L2 регуляризация весов листьев и более плавная, чем регуляризация L1. [6]

Создадим модель бустинга.

```
xg_reg = xgb.XGBRegressor(objective ='reg:linear', colsample_bytree = 0.5, learning_rate = 0.2, max_depth = 7, alpha = 10, n_estimators = 75)

xg_reg.fit(x_train,y_train)
preds = xg_reg.predict(x_test)
```

Вычислим ошибку прогноза

```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE: %f" % (rmse))
```

RMSE: 6.109611

Ошибка составила 6%.

Визуализация деревьев XGBoost

```
import matplotlib.pyplot as plt


xgb.plot_tree(xg_reg,num_trees=0)
plt.rcParams['figure.figsize'] = [20, 15]
plt.show()
```

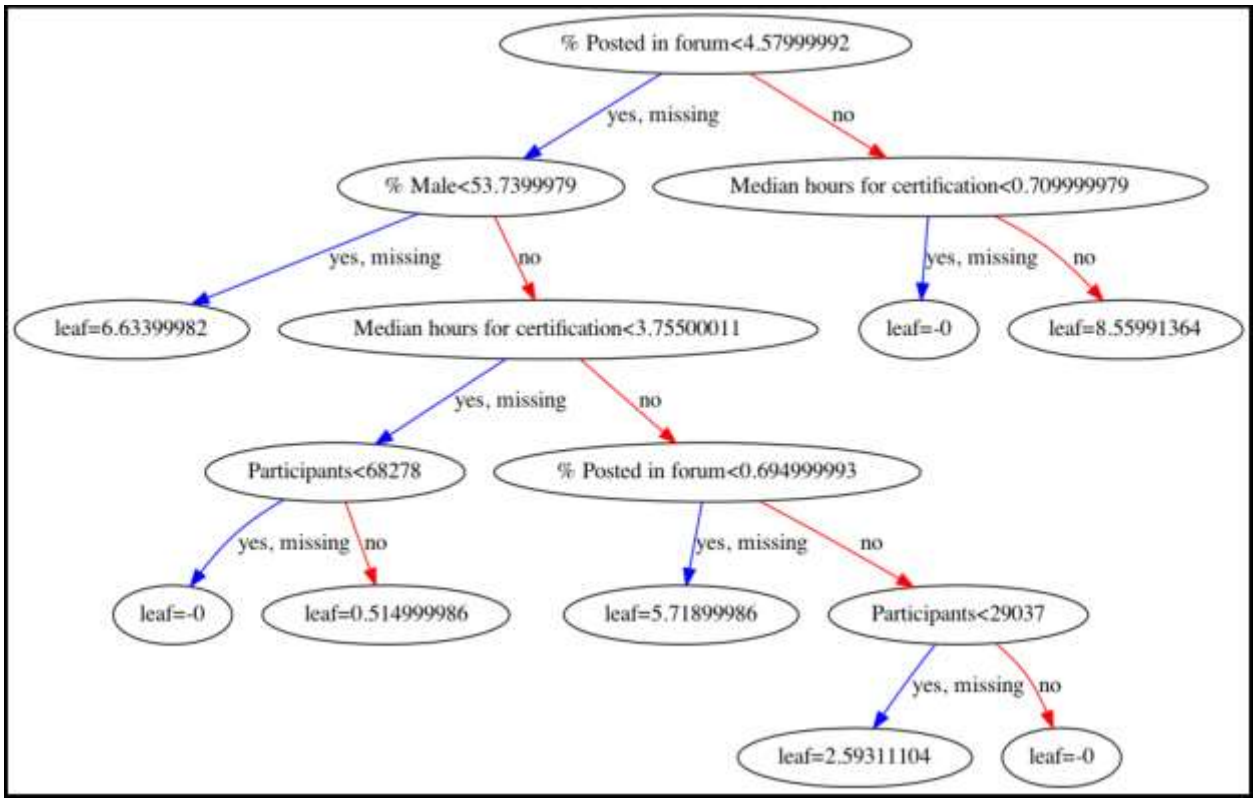

Рисунок 8. Деревья XGBoost

Построим график важности признаков на основе подобранных деревьев.

```
xgb.plot_importance(xg_reg)
plt.rcParams['figure.figsize'] = [15,15]
plt.show()
```

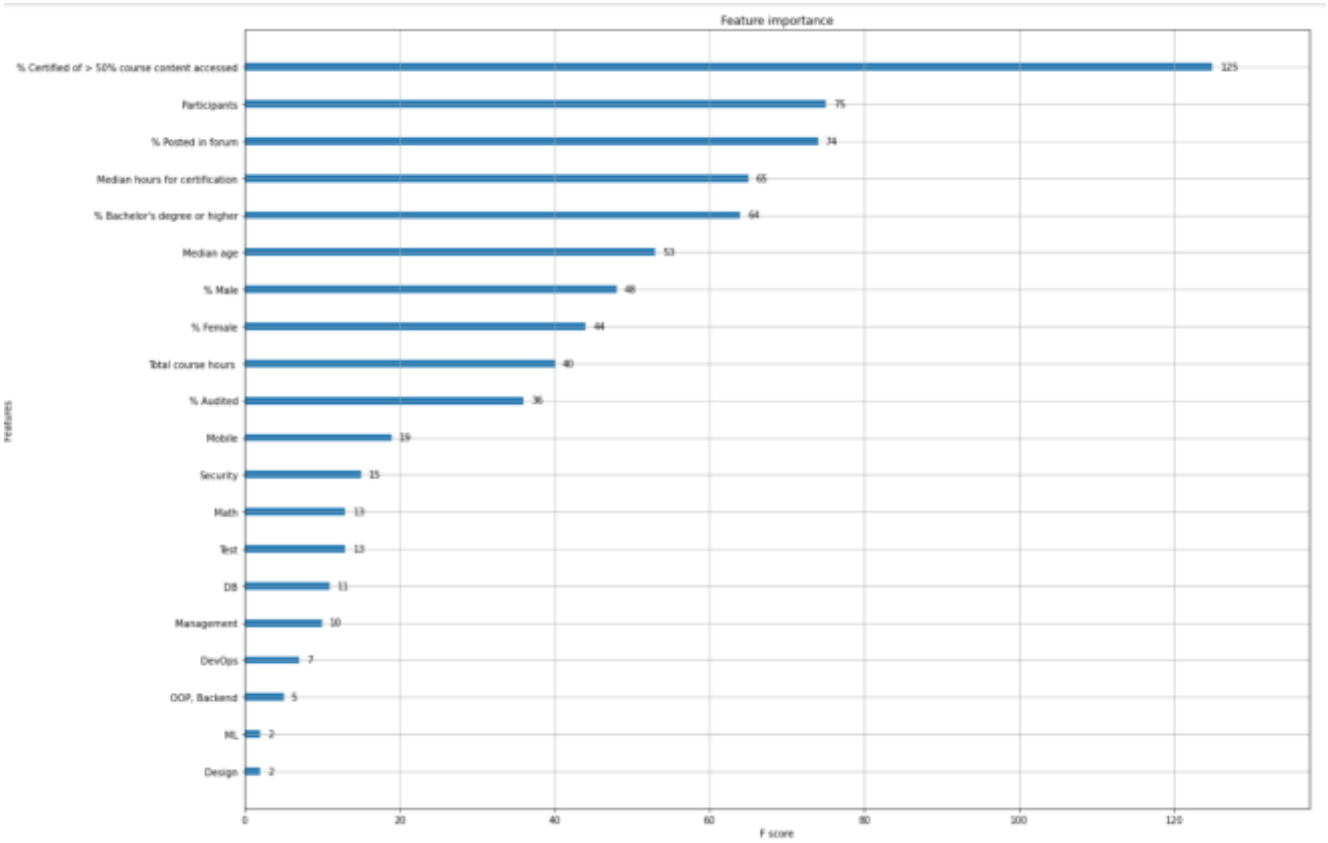

Рисунок 9. График важности признаков

По графику видно, что наибольшей важностью обладает признак «% Certified of > 50% course content accessed»

В данной статье была произведена очистка, визуализация, а также анализ данных от одной из платформ онлайн-курсов. С использованием библиотеки XGBoost была произведена попытка создания модели, которая предсказывает, завершит ли пользователь более 50% курса или нет. Точность прогноза модели составила около 94%.

Список использованных источников

1. Seaborn documentation. Режим доступа: https://seaborn.pydata.org/docs/ Дата обращения: 28.11.2020
2. Pandas documentation Режим доступа: https://pandas.pydata.org/docs/ Дата обращения: 28.11.2020
3. Введение в pandas: анализ данных на Python. Режим доступа: https://khashtamov.com/ru/pandas-introduction/ Дата обращения: 28.11.2020

4. Friedman J. Greedy Function Approximation: A Gradient Boosting Ma-chine. — IMS 1999 Reitz Lecture.
5. Nonita Sharma, XGBoost. The Extreme Gradient Boosting for Mining Ap-plications. - 2018 GRIN Verlag
6. Xgboost documentation. Режим доступа: https://xgboost.readthedocs.io/en/latest/ Дата обращения: 28.11.2020

Literature

1. Seaborn documentation. Access mode: https://seaborn.pydata.org/docs/ Date accessed: 11/28/2020
2. Pandas documentation Access mode: https://pandas.pydata.org/docs/ Date of access: 11/28/2020
3. Introduction to pandas: data analysis in Python. Access mode: https://khashtamov.com/ru/pandas-introduction/ Date of access: 11/28/2020
4. Friedman J. Greedy Function Approximation: A Gradient Boosting Machine. - IMS 1999 Reitz Lecture.
5. Nonita Sharma, XGBoost. The Extreme Gradient Boosting for Mining Ap-plications. - 2018 GRIN Verlag
6. Xgboost documentation. Access mode: https://xgboost.readthedocs.io/en/latest/ Date accessed: 28.11.2020