

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА - Российский технологический университет» РТУ МИРЭА**

**Институт** Информационных Технологий

**Кафедра** Вычислительной техники

**ПРАКТИЧЕСКАЯ РАБОТА №1**

**по дисциплине**

**«Проектирование интеллектуальных систем (Часть 1/2)»**

Студент группы ИКБО-04-22 Сариков Т.А.

*(Ф.И.О. студента)*

Руководитель работы Холмогоров В.В.

*(Ф.И.О. преподавателя)*

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc10014)

[1 ПОСТАНОВКА ЗАДАЧИ 4](#_Toc10015)

[2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc10016)

[2.1 Алгоритм Apriori 6](#_Toc10017)

[2.4 Генерация ассоциативных правил 8](#_Toc10020)

[3 ОПИСАНИЕ ДАННЫХ 9](#_Toc10021)

[3.1 Генерация данных 9](#_Toc10022)

[4 ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_Toc10023)

[4.1 Реализация алгоритма Apriori 11](#_Toc10024)

[ЗАКЛЮЧЕНИЕ 17](#_Toc10027)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc10028)

[ПРИЛОЖЕНИЯ 19](#_Toc10029)

# 

# ВВЕДЕНИЕ

Задача предсказания классов (классификация) является одной из наиболее востребованных. В данной практической работе рассматривается набор методов предобработки данных, а также обучение и сравнение двух моделей классификации: логистической регрессии и градиентного бустинга. Исходный датасет представляет собой данные об уровне удовлетворенности клиентов авиакомпании, а цель состоит в том, чтобы на основе набора признаков предсказывать, удовлетворён ли клиент. В отчёте подробно описана постановка задачи, теоретические основы используемых методов, характеристика данных, а также поэтапное выполнение практической части с соответствующими результатами и визуализациями.

# 1 ПОСТАНОВКА ЗАДАЧИ

Классификация — задача машинного обучения, в которой объектам (записям) ставятся в соответствие заранее определённые метки (классы). В данной работе переменная-метка называется satisfaction и принимает два значения:

* satisfied — удовлетворён;
* neutral or dissatisfied — нейтрален или не удовлетворён.

Обозначим выборку из объектов:

где — число признаков, а — бинарная метка (например, — «удовлетворён», — «не удовлетворён»).

# 2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В теоретической части приведены основные понятия и формулы, лежащие в основе методов, реализованных в практической части.

## 2.2 Оценка качества классификации

Для оценки качества бинарных классификаторов используют следующие метрики:

Accuracy - Доля правильно классифицированных объектов:

Где TP — число истинно положительных (правильно предсказанных «удовлетворённых»), TN — число истинно отрицательных (правильно предсказанных «неудовлетворённых»), — число ложноположительных (когда модель предсказала «удовлетворён», но на самом деле «неудовлетворён»), — число ложноотрицательных (когда модель предсказала «неудовлетворён», но на самом деле «удовлетворён»).

Precision (точность положительных предсказаний) - Доля истинных положительных среди всех объектов, предсказанных как положительные:

Recall (полнота или чувствительность) - Доля истинных положительных, которые модель правильно нашла:

F1-score - Гармоническое среднее между Precision и Recall:

Чем выше F1, тем сбалансированнее точность и полнота.

## 2.3 Обработка пропущенных значений

Если в датасете содержатся пропущенные () значения, их необходимо корректно обработать. В работе использован метод замены пропусков на медиану для числовых признаков:

Медиана устойчива к выбросам и позволяет избежать сильного влияния редких экстремальных значений.

Для категориальных признаков используется замена на наиболее частое значение (most frequent):

## 2.4 Логистическая регрессия

**Логистическая регрессия** — базовый линейный классификатор, строящий модель вероятности принадлежности к положительному классу по формуле:

где

— вектор признаков,

— вектор весов (коэффициентов),

— смещение (bias),

— сигмоидальная функция.

Оценка параметров производится путём минимизации логистической функции потерь на обучающей выборке

## 2.5 Градиентный бустинг (CatBoost)

Градиентный бустинг — метод ансамблевой классификации, строящий последовательность слабых моделей (как правило, деревьев решений), каждая из которых обучается на остатках ошибок предыдущей. Для задачи классификации с бинарной кросс-энтропией приближённая схема такова:

1. Инициализируется предсказанием-константой:

* где — функция потерь (Logloss).

1. На каждом шаге строится новое дерево , приближающее градиент функции потерь по текущим предсказаниям:

* затем дерево обучается, чтобы приблизить . Получаем модель .

1. Обновление ансамбля:

* где — скорость обучения (learning rate).

В результате получается модель

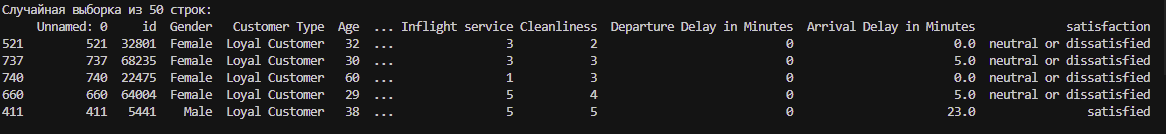
# 3 ОПИСАНИЕ ДАННЫХ

## 3.1 Анализ данных

Датасет был взят с платформы Kaggle

Датасет содержит информацию о клиентских отзывах и деталях их путешествий, задача предсказать будет ли удовлетворён пассажир полетом.

Случайные 5 строк датасета



**Рисунок 3.1.1 — семплы датасета**

Распределение некоторых категориальных фичей

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 3.1.2 — распределение категориальных признаков**

Распределение таргета. В этой работе проводится бинарная классификация (1: satisfied, 0: нейтрально или неуд)

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 3.1.3 — распределение таргета**

# 4 ПРАКТИЧЕСКАЯ ЧАСТЬ

В данной работе стараемся улучшить метрику F1.

## 4.1 Реализация алгоритма Apriori

После обучения бустинга получаем следующие результаты. После 500 итерации срабатывает раняя остановка

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 4.1.1 — Обучение кэтбуста**

Видим важности признаков по убыванию:

Изображение выглядит как текст, Шрифт, снимок экрана, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 4.1.2 — Обучение кэтбуста**

Замечаем, что наличие вай-фая и цель поездки сильно влияют на итоговое впечатление от поездки

Также видно, что градиентный бустинг лучше классифицирует чем лог рег

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

# ЗАКЛЮЧЕНИЕ

Сравнение различных моделей машинного обучения показало, что наилучшие результаты достигаются при использовании модели CatBoost и ансамбля моделей с точностью более 90%. Оптимизация по метрике F1 и настройка порога принятия решения позволили дополнительно улучшить качество классификации. Анализ важности признаков выявил, что ключевыми факторами, влияющими на удовлетворенность пассажиров, являются качество онлайн-регистрации, тип путешествия, класс обслуживания, комфорт сидений и развлечения на борту, в то время как задержки рейсов и демографические характеристики оказались менее значимыми. Полученные результаты могут быть использованы авиакомпанией для повышения качества обслуживания в наиболее важных для пассажиров аспектах.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Azevedo, P. J., Jorge, A. M. Comparing Rule Measures for Predictive Association Rules 2007.

# ПРИЛОЖЕНИЯ

Приложение А — Реализация обучения классификаторов.

**Приложение А**

Реализация обучения классификаторов

*Листинг А.1 — Генерация данных*

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split, cross\_val\_score, GridSearchCV

from sklearn.preprocessing import StandardScaler, OneHotEncoder, LabelEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

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

from sklearn.linear\_model import ElasticNet

from catboost import CatBoostRegressor, Pool

import warnings

warnings.filterwarnings('ignore')

RANDOM\_STATE = 42

np.random.seed(RANDOM\_STATE)

def load\_data(file\_path):

print("Загрузка и анализ данных...")

data = pd.read\_csv(file\_path)

print(f"Форма датасета: {data.shape}")

return data

def analyze\_data(data):

print("\n--- Первые 5 строк датасета ---")

print(data.head())

print("\n--- Информация о датасете ---")

print(data.info())

print("\n--- Статистика по числовым признакам ---")

print(data.describe())

print("\n--- Проверка пропущенных значений ---")

print(data.isnull().sum())

print("\n--- Уникальные значения категориальных признаков ---")

for col in ['sex', 'smoker', 'region']:

print(f"{col}: {data[col].unique()}")

return data

def visualize\_data(data):

print("\nСоздание визуализаций...\n")

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)

sns.histplot(data['charges'], kde=True)

plt.title('Распределение charges')

plt.subplot(2, 3, 2)

sns.histplot(np.log1p(data['charges']), kde=True)

plt.title('Log-распределение charges')

plt.subplot(2, 3, 3)

sns.scatterplot(x='age', y='charges', hue='smoker', data=data)

plt.title('Возраст vs Charges')

plt.subplot(2, 3, 4)

sns.scatterplot(x='bmi', y='charges', hue='smoker', data=data)

plt.title('ИМТ vs Charges')

plt.subplot(2, 3, 5)

sns.boxplot(x='smoker', y='charges', data=data)

plt.title('Charges в зависимости от курения')

plt.subplot(2, 3, 6)

sns.boxplot(x='region', y='charges', data=data)

plt.title('Charges в зависимости от региона')

plt.tight\_layout()

plt.savefig('charges\_visualization.png')

plt.close()

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

numeric\_cols = ['age', 'bmi', 'children', 'charges']

corr\_matrix = data[numeric\_cols].corr()

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Корреляционная матрица (числовые признаки)')

plt.savefig('correlation\_matrix.png')

plt.close()

print("Визуализации созданы и сохранены как изображения.")

def preprocess\_data(data):

print("\nПредобработка данных...")

data\_processed = data.copy()

categorical\_features = ['sex', 'smoker', 'region']

label\_encoders = {}

for col in categorical\_features:

le = LabelEncoder()

data\_processed[col] = le.fit\_transform(data\_processed[col])

label\_encoders[col] = le

print(f"Кодирование {col}: {dict(zip(le.classes\_, le.transform(le.classes\_)))}")

X = data\_processed.drop('charges', axis=1)

y = data\_processed['charges']

if data['charges'].skew() > 1:

print(f"Целевая переменная имеет скошенное распределение (skew={data['charges'].skew():.2f}), применяем логарифмирование")

y = np.log1p(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=RANDOM\_STATE

)

print(f"Размер обучающей выборки: {X\_train.shape}")

print(f"Размер тестовой выборки: {X\_test.shape}")

return X\_train, X\_test, y\_train, y\_test, y.skew() > 1

def train\_models(X\_train, y\_train, X\_test, y\_test, is\_log\_transformed):

results = {}

print("\n--- Обучение модели CatBoost ---")

catboost\_model, catboost\_y\_test, catboost\_y\_pred = train\_catboost(X\_train, y\_train, X\_test, y\_test, is\_log\_transformed)

results['CatBoost'] = {

'model': catboost\_model,

'y\_test': catboost\_y\_test,

'y\_pred': catboost\_y\_pred

}

print("\n--- Обучение модели ElasticNet ---")

elastic\_model, elastic\_y\_test, elastic\_y\_pred = train\_elastic(X\_train, y\_train, X\_test, y\_test, is\_log\_transformed)

results['ElasticNet'] = {

'model': elastic\_model,

'y\_test': elastic\_y\_test,

'y\_pred': elastic\_y\_pred

}

print("\n--- Сравнение моделей ---")

compare\_models(results)

return catboost\_model, catboost\_y\_test, catboost\_y\_pred

def train\_catboost(X\_train, y\_train, X\_test, y\_test, is\_log\_transformed):

print("Обучение модели CatBoost...")

numerical\_features = ['age', 'bmi', 'children']

categorical\_features = ['sex', 'smoker', 'region']

catboost\_model = CatBoostRegressor(

iterations=500,

learning\_rate=0.1,

depth=6,

loss\_function='RMSE',

eval\_metric='RMSE',

random\_seed=RANDOM\_STATE,

cat\_features=categorical\_features,

verbose=100

)

catboost\_model.fit(

X\_train, y\_train,

eval\_set=(X\_test, y\_test),

early\_stopping\_rounds=50

)

y\_pred = catboost\_model.predict(X\_test)

if is\_log\_transformed:

y\_test\_original = np.expm1(y\_test)

y\_pred\_original = np.expm1(y\_pred)

else:

y\_test\_original = y\_test

y\_pred\_original = y\_pred

rmse = np.sqrt(mean\_squared\_error(y\_test\_original, y\_pred\_original))

mae = mean\_absolute\_error(y\_test\_original, y\_pred\_original)

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

print(f"\nМетрики CatBoost на тестовой выборке:")

print(f"RMSE: {rmse:.2f}")

print(f"MAE: {mae:.2f}")

print(f"R^2: {r2:.4f}")

return catboost\_model, y\_test\_original, y\_pred\_original

def train\_elastic(X\_train, y\_train, X\_test, y\_test, is\_log\_transformed):

print("Обучение модели ElasticNet...")

numerical\_features = ['age', 'bmi', 'children']

categorical\_features = ['sex', 'smoker', 'region']

preprocessor = ColumnTransformer(

transformers=[

('num', StandardScaler(), numerical\_features),

('cat', OneHotEncoder(drop='first'), categorical\_features)

]

)

param\_grid = {

'elasticnet\_\_alpha': [0.001, 0.01, 0.1, 1.0, 10.0],

'elasticnet\_\_l1\_ratio': [0.1, 0.3, 0.5, 0.7, 0.9]

}

elastic\_pipeline = Pipeline([

('preprocessor', preprocessor),

('elasticnet', ElasticNet(random\_state=RANDOM\_STATE, max\_iter=10000))

])

grid\_search = GridSearchCV(

elastic\_pipeline,

param\_grid,

cv=5,

scoring='neg\_mean\_squared\_error',

n\_jobs=-1

)

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

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

print(f"Лучшие параметры ElasticNet: {grid\_search.best\_params\_}")

y\_pred = elastic\_model.predict(X\_test)

if is\_log\_transformed:

y\_test\_original = np.expm1(y\_test)

y\_pred\_original = np.expm1(y\_pred)

else:

y\_test\_original = y\_test

y\_pred\_original = y\_pred

rmse = np.sqrt(mean\_squared\_error(y\_test\_original, y\_pred\_original))

mae = mean\_absolute\_error(y\_test\_original, y\_pred\_original)

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

print(f"\nМетрики ElasticNet на тестовой выборке:")

print(f"RMSE: {rmse:.2f}")

print(f"MAE: {mae:.2f}")

print(f"R^2: {r2:.4f}")

return elastic\_model, y\_test\_original, y\_pred\_original

def compare\_models(results):

model\_names = list(results.keys())

rmse\_scores = []

mae\_scores = []

r2\_scores = []

for name in model\_names:

y\_test = results[name]['y\_test']

y\_pred = results[name]['y\_pred']

rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

mae = mean\_absolute\_error(y\_test, y\_pred)

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

rmse\_scores.append(rmse)

mae\_scores.append(mae)

r2\_scores.append(r2)

comparison\_df = pd.DataFrame({

'Модель': model\_names,

'RMSE': rmse\_scores,

'MAE': mae\_scores,

'R^2': r2\_scores

})

print("\nСравнение метрик моделей:")

print(comparison\_df)

plt.figure(figsize=(12, 10))

plt.subplot(3, 1, 1)

sns.barplot(x='Модель', y='RMSE', data=comparison\_df)

plt.title('Сравнение RMSE')

plt.ylabel('RMSE (меньше лучше)')

plt.subplot(3, 1, 2)

sns.barplot(x='Модель', y='MAE', data=comparison\_df)

plt.title('Сравнение MAE')

plt.ylabel('MAE (меньше лучше)')

plt.subplot(3, 1, 3)

sns.barplot(x='Модель', y='R^2', data=comparison\_df)

plt.title('Сравнение R^2')

plt.ylabel('R^2 (больше лучше)')

plt.tight\_layout()

plt.savefig('model\_comparison.png')

plt.close()

print("График сравнения моделей сохранен как 'model\_comparison.png'.")

def analyze\_results(model, X\_train, y\_test, y\_pred):

print("\nАнализ результатов и важности признаков...")

feature\_importance = model.get\_feature\_importance()

features = X\_train.columns

plt.figure(figsize=(10, 6))

sorted\_idx = np.argsort(feature\_importance)

plt.barh(range(len(sorted\_idx)), feature\_importance[sorted\_idx])

plt.yticks(range(len(sorted\_idx)), [features[i] for i in sorted\_idx])

plt.title('Важность признаков в модели CatBoost')

plt.savefig('feature\_importance.png')

plt.close()

plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_pred, alpha=0.5)

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], 'r--')

plt.xlabel('Фактические значения')

plt.ylabel('Предсказанные значения')

plt.title('Предсказанные vs Фактические значения')

plt.savefig('prediction\_vs\_actual.png')

plt.close()

errors = y\_test - y\_pred

plt.figure(figsize=(10, 6))

sns.histplot(errors, kde=True)

plt.title('Распределение ошибок')

plt.savefig('error\_distribution.png')

plt.close()

print("Графики сохранены как изображения.")

def main(file\_path):

data = load\_data(file\_path)

data = analyze\_data(data)

visualize\_data(data)

X\_train, X\_test, y\_train, y\_test, is\_log\_transformed = preprocess\_data(data)

model, y\_test\_original, y\_pred\_original = train\_models(X\_train, y\_train, X\_test, y\_test, is\_log\_transformed)

analyze\_results(model, X\_train, y\_test\_original, y\_pred\_original)

print("\nЗадача выполнена успешно!")

if \_\_name\_\_ == "\_\_main\_\_":

main('expenses.csv')