Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Омский государственный технический университет»

Кафедра «Автоматизированные системы обработки информации и управления»

Курсовой проект
по дисциплине
«Динамические языки программирования»

Выполнил:
Студент гр. ПИН-202
Галинуров А.Е. _____
(подп., дата)

Проверил:
Доцент каф. АСОИУ
Кабанов А.А. _____
(подп., дата)

Омск 2023

Реферат

ОТЧЕТ 27 с., 22 рис., 3 источника.

КУРСОВОЙ ПРОЕКТ, МАШИННОЕ ОБУЧЕНИЕ

Цель курсовой работы — ознакомиться и приобрести базовые знания в области машинного обучения.

В данной работе выполнены:

1. метод к- ближайших соседей;
2. метод машины опорных векторов;
3. методы линейной и логистической регрессий;
4. метод наивного Байеса;
5. методы решающего дерева и случайного леса;
6. метод CatBoost.

СОДЕРЖАНИЕ

Введение

Машинное обучение — одна из наиболее динамично развивающихся областей, которая позволяет компьютерам обучаться на основе данных и делать прогнозы или принимать решения без явного программирования.

Цель настоящей курсовой работы заключается в ознакомлении с основами машинного обучения и рассмотрении различных методов классификации. В ходе исследования были проведены эксперименты с различными алгоритмами, такими как метод к-ближайших соседей, машина опорных векторов, линейная и логистическая регрессии, наивный Байес, а также решающее дерево и случайный лес.

В рамках данной работы представлен обзор и сравнение указанных методов классификации на простых модельных данных. Каждый из методов исследован с целью понимания их принципов работы, преимуществ и ограничений.

1. Метод к-ближайших соседей (k-NN)

1.1 Описание метода

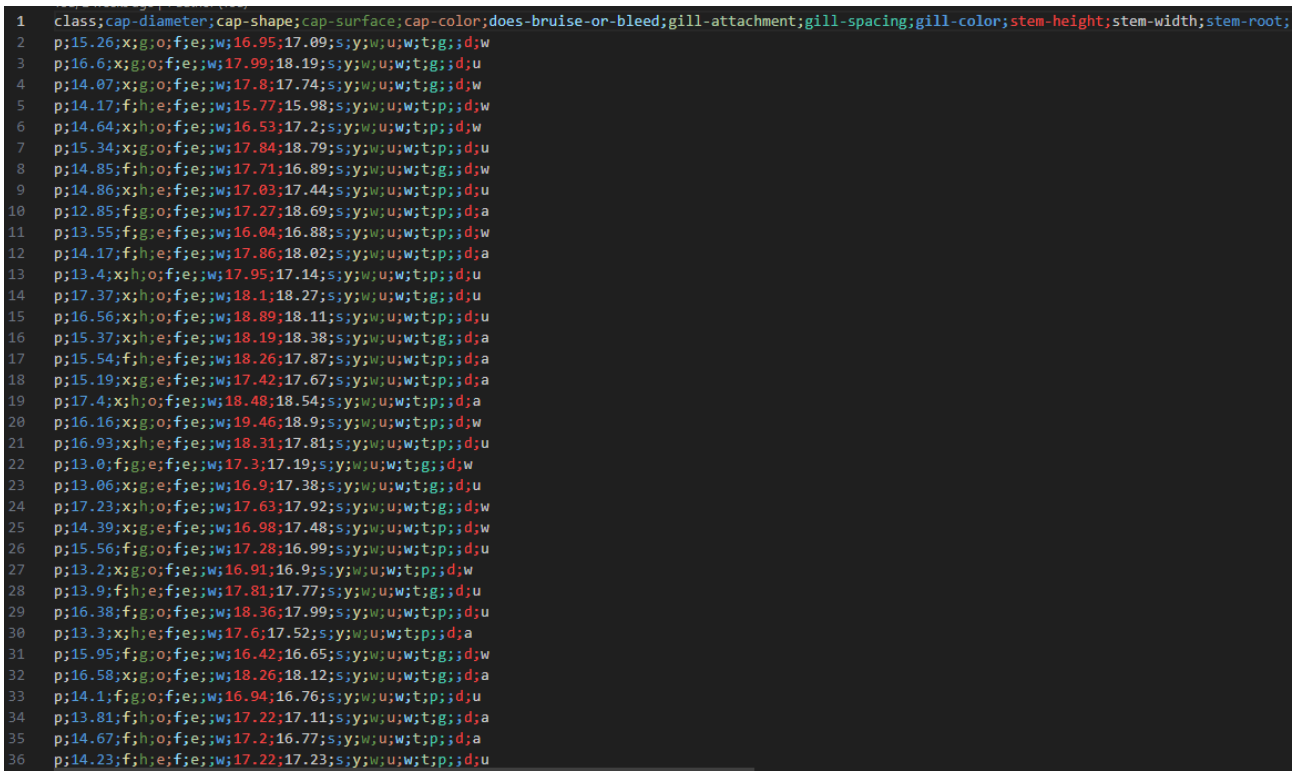
Метод к-ближайших соседей (k-NN) относится к одному из простейших алгоритмов классификации в машинном обучении. Он основывается на принципе близости объектов: если у объекта есть соседи известного класса, то скорее всего, этот объект также принадлежит к этому классу.

1.2 Принцип работы

При классификации нового объекта метод k-NN ищет k ближайших к нему объектов в обучающем наборе данных. Затем присваивает новому объекту тот класс, который наиболее часто встречается среди его соседей.

1.3 Применение на модельных данных

На рисунке 1 представлен датасет, на основе которого будут применяться все представленные методы:



```
class;cap-diameter;cap-shape;cap-surface;cap-color;does-bruise-or-bleed;gill-attachment;gill-spacing;gill-color;stem-height;stem-width;stem-root;
p;15.26;x;g;o;f;e;;w;16.95;17.09;s;y;w;u;w;t;g;;d;w
p;16.6;x;g;o;f;e;;w;17.99;18.19;s;y;w;u;w;t;g;;d;u
p;14.07;x;g;o;f;e;;w;17.8;17.74;s;y;w;u;w;t;g;;d;w
p;14.17;f;h;e;f;e;;w;15.77;15.98;s;y;w;u;w;t;p;;d;w
p;14.64;x;h;o;f;e;;w;16.53;17.2;s;y;w;u;w;t;p;;d;w
p;15.34;x;g;o;f;e;;w;17.84;18.79;s;y;w;u;w;t;p;;d;u
p;14.85;f;h;o;f;e;;w;17.71;16.89;s;y;w;u;w;t;g;;d;w
p;14.86;x;h;e;f;e;;w;17.03;17.44;s;y;w;u;w;t;p;;d;u
p;12.85;f;g;o;f;e;;w;17.27;18.69;s;y;w;u;w;t;p;;d;a
p;13.55;f;g;e;f;e;;w;16.04;16.88;s;y;w;u;w;t;p;;d;w
p;14.17;f;h;e;f;e;;w;17.86;18.02;s;y;w;u;w;t;p;;d;a
p;13.4;x;h;o;f;e;;w;17.95;17.14;s;y;w;u;w;t;p;;d;u
p;17.37;x;h;o;f;e;;w;18.1;18.27;s;y;w;u;w;t;g;;d;u
p;16.56;x;h;o;f;e;;w;18.89;18.11;s;y;w;u;w;t;p;;d;u
p;15.37;x;h;e;f;e;;w;18.19;18.38;s;y;w;u;w;t;g;;d;a
p;15.54;f;h;e;f;e;;w;18.26;17.87;s;y;w;u;w;t;p;;d;a
p;15.19;x;g;e;f;e;;w;17.42;17.67;s;y;w;u;w;t;p;;d;a
p;17.4;x;h;o;f;e;;w;18.48;18.54;s;y;w;u;w;t;p;;d;a
p;16.16;x;g;o;f;e;;w;19.46;18.9;s;y;w;u;w;t;p;;d;w
p;16.93;x;h;e;f;e;;w;18.31;17.81;s;y;w;u;w;t;p;;d;u
p;13.0;f;g;e;f;e;;w;17.3;17.19;s;y;w;u;w;t;g;;d;w
p;13.06;x;g;e;f;e;;w;16.9;17.38;s;y;w;u;w;t;g;;d;u
p;17.23;x;h;o;f;e;;w;17.63;17.92;s;y;w;u;w;t;g;;d;w
p;14.39;x;g;e;f;e;;w;16.98;17.48;s;y;w;u;w;t;p;;d;w
p;15.56;f;g;o;f;e;;w;17.28;16.99;s;y;w;u;w;t;p;;d;u
p;13.2;x;g;o;f;e;;w;16.91;16.9;s;y;w;u;w;t;p;;d;w
p;13.9;f;h;e;f;e;;w;17.81;17.77;s;y;w;u;w;t;g;;d;u
p;16.38;f;g;o;f;e;;w;18.36;17.99;s;y;w;u;w;t;p;;d;u
p;13.3;x;h;e;f;e;;w;17.6;17.52;s;y;w;u;w;t;p;;d;a
p;15.95;f;g;o;f;e;;w;16.42;16.65;s;y;w;u;w;t;g;;d;w
p;16.58;x;g;o;f;e;;w;18.26;18.12;s;y;w;u;w;t;g;;d;a
p;14.1;f;g;o;f;e;;w;16.94;16.76;s;y;w;u;w;t;p;;d;u
p;13.81;f;h;o;f;e;;w;17.22;17.11;s;y;w;u;w;t;g;;d;a
p;14.67;f;h;o;f;e;;w;17.2;16.77;s;y;w;u;w;t;p;;d;a
p;14.23;f;h;e;f;e;;w;17.22;17.23;s;y;w;u;w;t;p;;d;u
```

Рисунок 1 – Датасет

Набор данных смоделированных грибов для бинарной классификации на съедобные и ядовитые. В каждой строке есть информация о грибе, представленная различными атрибутами:

1. **CAP-DIAMETER**: диаметр колпачка (m): число плавучести в см.
2. **CAP-SHAPE**: форма крышки (n): колокольчатая=b, коническая=c, выпуклая=x, плоская=f, утопленная=s, сферическая=p, другие=o
3. **CAP-SURFACE**: поверхность шляпки (n): волокнистая=i, бороздки=g, чешуйчатая=y, гладкая=s, блестящая=h, кожистая=l, шелковистая=k, липкая=t, морщинистый=w, мясистый=e
4. **CAP-COLOR**: цвет шапки (n): коричневый=n, буфф=b, серый=g, зеленый=r, розовый=p, фиолетовый=u, красный=e, белый=w, желтый=y, синий=l, оранжевый=o, черный=k
5. **DOES-BRUISE-BLEED**: кровоточит ли синяк (n): синяки или кровотечение=t, нет=f
6. **GILL-ATTACHMENT**: жаберный аппарат (n): аднатный=a, аднексированный=x, декуррентный=d, свободный=e, синуат=c, поры=p, нет=ф, неизвестно=?
7. **GILL-SPACING** расстояние между жабрами (n): близко=c, далеко=d, нет=f
8. **GILL-COLOR**: цвет жабр (n): см. цвет крышечки + нет=f
9. **STEM-HEIGHT**: высота стебля (м): плавающее число в см.
10. **STEM-WIDTH**: ширина стебля (м): плавающее число в мм
11. **STEM-ROOT**: стеблекорень (n): луковичный=b, вздутый=s, булавовидный=c, чашевидный=u, равный=e, ризоморфы=z, корневище=r
12. **STEM-SURFACE**: поверхность стебля (n): см. поверхность шапки + нет=f
13. **STEM-COLOR**: цвет стебля (n): см. цвет шапки + none=f
14. **VEIL-TYPE**: тип вуали (n): частичная=p, универсальная=u

15. **VEIL-COLOR**: вуаль-цвет (n): см. колпачок-цвет + none=f
16. **HAS-RING**: имеет ли кольцо? Кольцо=t, нет=f
17. **RING-TYPE**: тип кольца (n): паутинистое=c, эмансипирующее=e, вспыхивающее=r, рифленое=g, большое=l, подвесное=p, оболочка=s, зональное=z, чешуйчатое=y, подвижное=m, нет=f, неизвестное=?
18. **SPORT-PRINT-COLOR**: цвет отпечатка споры (n): см. цвет колпачка
19. **HABITAT**: среда обитания (n): травы=g, листья=l, луга=m, тропинки=p, пустоши=h, городские=u, отходы=w, леса=d
20. **SEASON**: сезон (n): весна=s, лето=u, осень=a, зима=w
21. **CLASS:** целевая переменная, где p указывается на то, что гриб является ядовитым, а e съедобным.

На рисунках 2–4 представлена подготовка данных

```
frame = pd.read_csv('data.csv', sep=';', encoding='cp1252')
predictors = ['cap-diameter', 'stem-height']
outcome = 'class'
frame
```

Python

	class	cap-diameter	cap-shape	cap-surface	cap-color	does-bruise-or-bleed	gill-attachment	gill-spacing	gill-color	stem-height	...	stem-root	stem-surface	stem-color	veil-type	veil-color	has-ring	ring-type	spore-print-color	habitat	season
0	p	15.26	x	g	o	f	e	NaN	w	16.95	...	s	y	w	u	w	t	g	NaN	d	w
1	p	16.60	x	g	o	f	e	NaN	w	17.99	...	s	y	w	u	w	t	g	NaN	d	u
2	p	14.07	x	g	o	f	e	NaN	w	17.80	...	s	y	w	u	w	t	g	NaN	d	w
3	p	14.17	f	h	e	f	e	NaN	w	15.77	...	s	y	w	u	w	t	p	NaN	d	w
4	p	14.64	x	h	o	f	e	NaN	w	16.53	...	s	y	w	u	w	t	p	NaN	d	w
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
61064	p	1.18	s	s	y	f	f	f	f	3.93	...	NaN	NaN	y	NaN	NaN	f	f	NaN	d	a
61065	p	1.27	f	s	y	f	f	f	f	3.18	...	NaN	NaN	y	NaN	NaN	f	f	NaN	d	a
61066	p	1.27	s	s	y	f	f	f	f	3.86	...	NaN	NaN	y	NaN	NaN	f	f	NaN	d	u
61067	p	1.24	f	s	y	f	f	f	f	3.56	...	NaN	NaN	y	NaN	NaN	f	f	NaN	d	u
61068	p	1.17	s	s	y	f	f	f	f	3.25	...	NaN	NaN	y	NaN	NaN	f	f	NaN	d	u

61069 rows × 21 columns

Рисунок 2 – Подготовка данных

```
frame.isnull().sum()
frame = frame.astype({
                      'class' : 'category',
                      'cap-shape' : 'category',
                      'cap-diameter' : 'float16',
                      'cap-surface' : 'category',
                      'cap-color' : 'category',
                      'does-bruise-or-bleed' : 'category',
                      'gill-attachment' : 'category',
                      'gill-spacing' : 'category',
                      'gill-color' : 'category',
                      'stem-height' : 'float16',
                      'stem-width' : 'float16',
                      'stem-root' : 'category',
                      'stem-surface' : 'category',
                      'stem-color' : 'category',
                      'veil-type' : 'category',
                      'veil-color' : 'category',
                      'has-ring' : 'category',
                      'ring-type' : 'category',
                      'spore-print-color' : 'category',
                      'habitat' : 'category',
                      'season' : 'category'
                      })
frame.dtypes
```

[3]

Рисунок 3 – Подготовка данных

```
encoder = LabelEncoder()
for column in frame.columns:
    if frame[column].dtype == 'category':
        frame[column] = encoder.fit_transform(frame[column])

frame.head()
```

Python

	class	cap-diameter	cap-shape	cap-surface	cap-color	does-bruise-or-bleed	gill-attachment	gill-spacing	gill-color	stem-height	...	stem-root	stem-surface	stem-color	veil-type	veil-color	has-ring	ring-type	spore-print-color	habitat	season
0	1	15.257812	6	2	6	0	2	3	10	16.953125	...	4	7	11	0	4	1	2	7	0	3
1	1	16.593750	6	2	6	0	2	3	10	17.984375	...	4	7	11	0	4	1	2	7	0	2
2	1	14.070312	6	2	6	0	2	3	10	17.796875	...	4	7	11	0	4	1	2	7	0	3
3	1	14.171875	2	3	1	0	2	3	10	15.773438	...	4	7	11	0	4	1	5	7	0	3
4	1	14.640625	6	3	6	0	2	3	10	16.531250	...	4	7	11	0	4	1	5	7	0	3

5 rows × 21 columns

Рисунок 4 – Подготовка данных

На рисунке 5 представлены разделение данных и нормализация

```
y = frame['class'].values
x = frame.drop(['class'], axis=1).values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

Рисунок 5 – Разделение данных и нормализация

На рисунке 6 представлены подбор гиперпараметров и наилучшего k

```
param_grid = {'n_neighbors': range(1, 20),
              'metric': ['euclidean', 'manhattan', 'chebyshev', 'minkowski']}
```

Чтобы найти лучшее значение для k, будем использовать инструмент под названием GridSearchCV. Это инструмент, который часто используется для настройки гиперпараметров моделей машинного обучения. В вашем случае это поможет автоматически найти наилучшее значение k для набора данных.

```
gridsearch = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5, scoring='accuracy')
gridsearch.fit(x_train, y_train)
gridsearch.best_params_
```

```
{'metric': 'manhattan', 'n_neighbors': 1}
```

Рисунок 6 – Подбор гиперпараметра и наилучшего k

На рисунке 7 представлены проверка на тестовых данных и вывод результатов

```
model = KNeighborsClassifier(n_neighbors=best_k, metric=best_metric)
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
print(f"Точность модели: {accuracy}")

report = classification_report(y_test, y_pred, zero_division=1)
print(f'Отчет о классификации : \n{report}')
```

```
Точность модели: 1.0
Отчет о классификации :
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5374
           1       1.00      1.00      1.00      6840

    accuracy                           1.00     12214
   macro avg       1.00      1.00      1.00     12214
weighted avg       1.00      1.00      1.00     12214
```

Рисунок 7 – Проверка на тестовых данных и вывод результатов

1.4 Преимущества и ограничения

Преимущества метода k-NN:

- Прост в реализации и понимании.
- Не требует обучения модели, пока поступают новые данные.
- Хорошо подходит для начальной оценки данных и быстрых прототипов.

Однако у метода k-NN есть и ограничения:

- Чувствителен к выбросам в данных.
- Неэффективен на больших наборах данных из-за вычислительной сложности.
- Не учитывает значимость признаков, все признаки равнозначны.

Метод k-NN полезен для первичного анализа данных, но в реальных приложениях может потребоваться более сложная модель для учета различных аспектов и повышения точности предсказаний.

2. Метод машины опорных векторов (SVM)

2.1 Описание метода

Метод машины опорных векторов (SVM) является мощным алгоритмом машинного обучения, используемым для задач классификации и регрессии. Основная идея заключается в поиске оптимальной разделительной гиперплоскости, которая максимально разделяет классы в данных.

2.2 Разделительная гиперплоскость и принцип работы

SVM строит гиперплоскость в n-мерном пространстве, где n - количество признаков. Эта гиперплоскость разделяет пространство на две части и максимизирует расстояние (зазор) между объектами разных классов, называемое отступом. Оптимальная гиперплоскость выбирается так, чтобы этот отступ был максимальным.

2.3 Применение на модельных данных

На рисунке 8 представлены гиперпараметры и подбор гиперпараметров с помощью перекрестной проверки

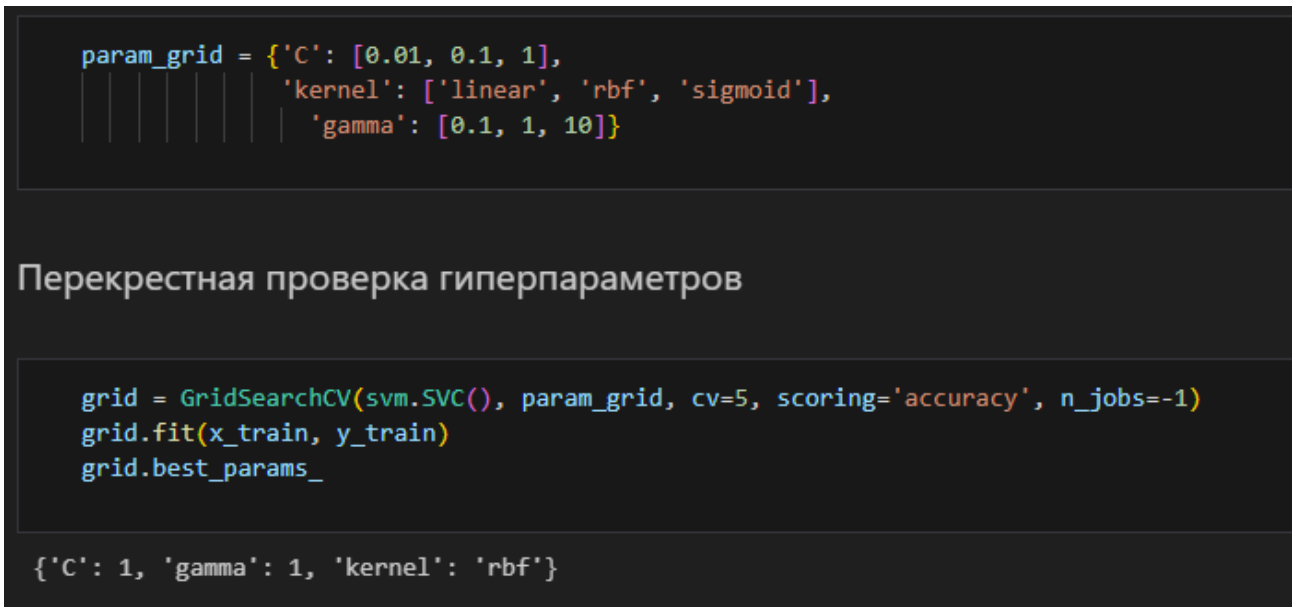


Рисунок 8 – Подбор гиперпараметров с помощью перекрестной проверки

Вывод лучших гиперпараметров и оценка производительности модели на тестовом наборе

```
best_params = {'kernel': grid.best_params_['kernel'],
               'C': grid.best_params_['C'],
               'gamma': grid.best_params_['gamma']}

model = svm.SVC(**best_params)
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Точность модели: {accuracy}')

report = classification_report(y_test, y_pred, zero_division=1)
print(report)

stratified_kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cross_val_scores = cross_val_score(model, x_train, y_train, cv=stratified_kfold)

print(f'Средняя точность перекрестной проверки: {cross_val_scores.mean()}')
```

```
Точность модели: 0.9998362534796136
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5374
           1       1.00      1.00      1.00      6840

    accuracy                           1.00     12214
   macro avg       1.00      1.00      1.00     12214
weighted avg       1.00      1.00      1.00     12214

Средняя точность перекрестной проверки: 0.9997748439259031
```

Рисунок 9 Оценка производительности модели натестовом наборе

2.4 Преимущества и ограничения

Преимущества метода SVM:

- Эффективен в пространствах с большим количеством признаков.
- Хорошо работает в условиях разделяемых данных с помощью нелинейных ядер.
- Стабилен при обучении на небольшом наборе данных.

Однако у метода SVM есть и ограничения:

- Требует тщательного выбора гиперпараметров и ядра.
- Неэффективен при работе с большими наборами данных из-за вычислительной сложности.
- Может быть чувствителен к выбросам в данных.

SVM - мощный алгоритм, который может быть эффективен при правильной настройке, но требует опыта для правильного применения и настройки гиперпараметров для достижения оптимальных результатов.

3. Методы линейной и логистической регрессии

3.1 Описание методов

Линейная регрессия – это метод, используемый для прогнозирования значений непрерывной зависимой переменной на основе линейной комбинации независимых переменных. Основная идея заключается в поиске линейной зависимости между предикторами и целевой переменной.

Применение линейной регрессии может быть полезным при прогнозировании численных значений, например, прогнозировании цены на недвижимость на основе её характеристик, таких как площадь, количество комнат и т.д.

Логистическая регрессия используется для задач классификации, прогнозируя вероятность принадлежности объекта к определенному классу. В отличие от линейной регрессии, логистическая регрессия применяется для бинарной или многоклассовой классификации.

3.2 Принцип работы

На рисунках 10 – 11 представлена работа с линейной регрессией.

```
grid_param = {'fit_intercept': [True, False]}
```

```
grid = GridSearchCV(LinearRegression(), grid_param, cv=5, n_jobs=-1)
grid.fit(x_train, y_train)
grid.best_params_
```

```
{'fit_intercept': True}
```

Рисунок 10 – Перекрестная проверка гиперпараметров линейной регрессии

```
best_fit = grid.best_params_['fit_intercept']
model = LinearRegression(fit_intercept=best_fit)
model.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
y_pred = model.predict(x_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("MAE: ", mae)
print("MSE: ", mse)
print("R^2: ", r2)
```

```
MAE:  0.43959386612180995
MSE:  0.2177016476007951
R^2:  0.11646494870130308
```

Рисунок 11 – Оценка точности модели на тестовых данных

На рисунках 12 – 13 представлена работа с логистической регрессией.

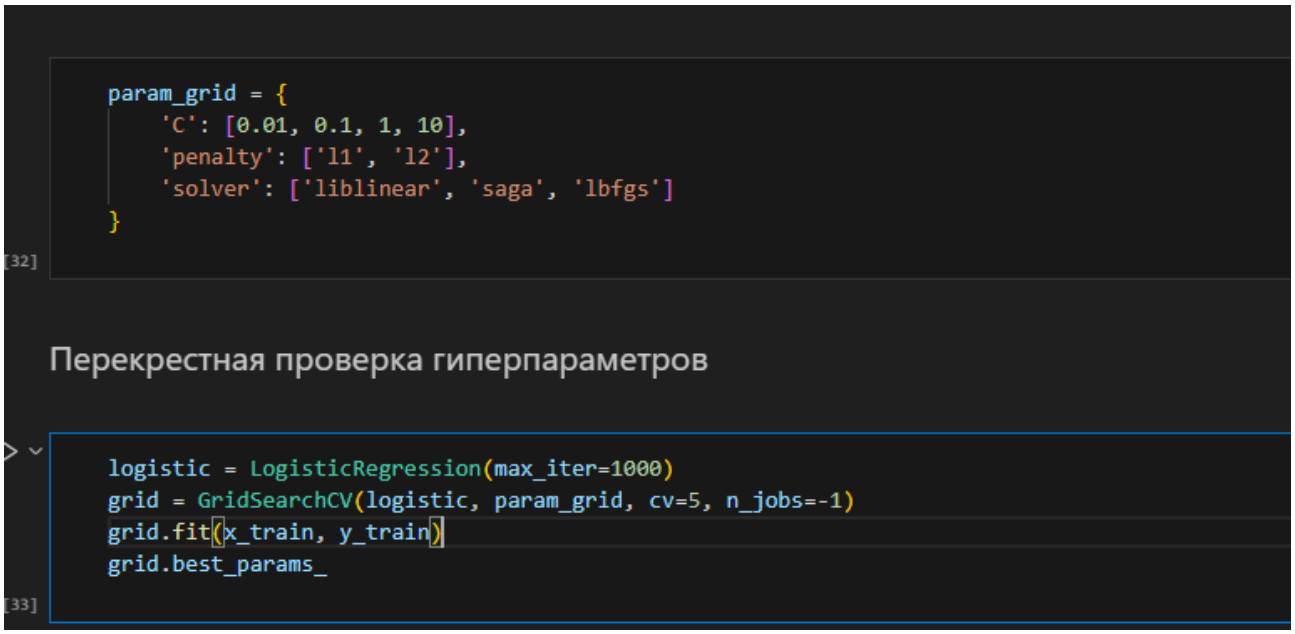


Рисунок 12 Перекрестная проверка гипермараметров логистической регрессии

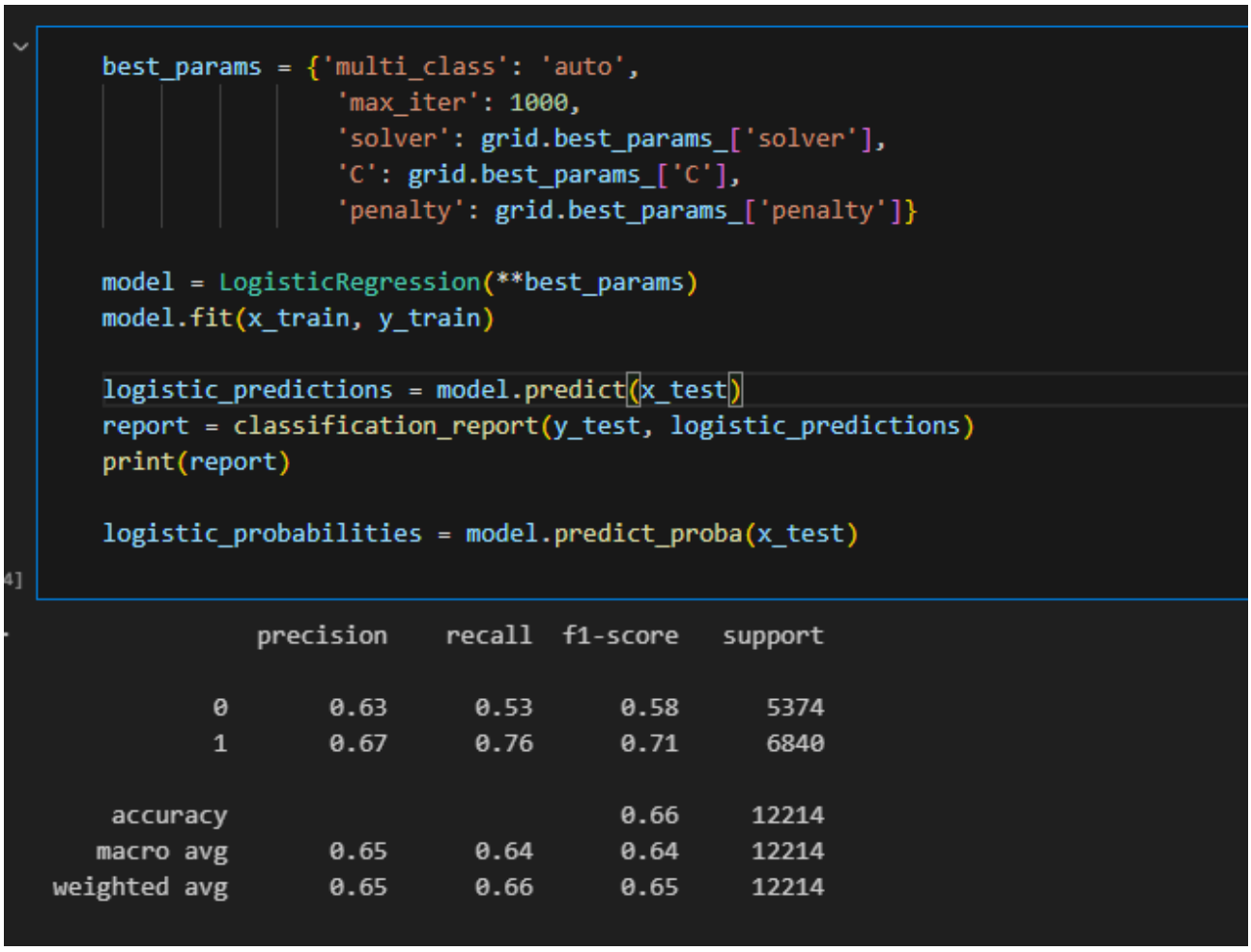


Рисунок 13 – Оценка точности модели на тестовых данных

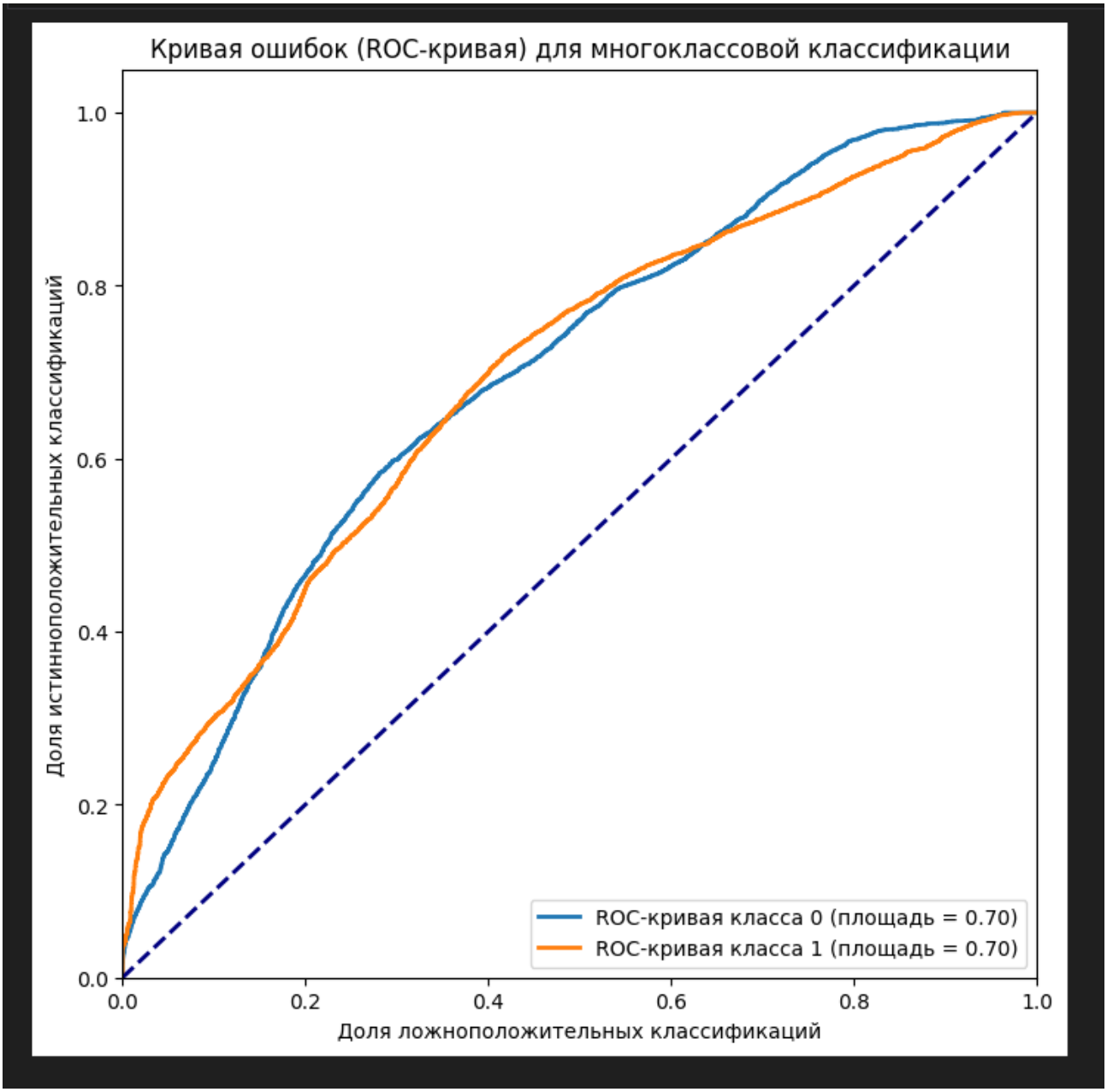


Рисунок 14 – Кривая ошибок для многоклассовой классификации

3.3 Сравнительный анализ с другими методами

Линейная и логистическая регрессии имеют свои сильные и слабые стороны по сравнению с другими методами. Линейная регрессия хорошо работает при предсказании непрерывных значений, тогда как логистическая регрессия применяется для задач классификации. Однако оба метода чувствительны к выбросам в данных и могут быть недостаточно гибкими для улавливания сложных нелинейных зависимостей. Поэтому в случае сложных данных и нелинейных связей эти методы могут оказаться менее эффективными по сравнению с другими алгоритмами, такими как деревья решений или нейронные сети.

4. Метод наивного Байеса

4.1 Описание метода

Метод наивного Байеса основан на теореме Байеса и предполагает независимость между признаками. Он использует вероятностный подход к классификации и основывается на простой предпосылке, что признаки объектов независимы между собой при условии принадлежности к определенному классу.

4.2 Применение на модельных данных

На рисунке 15 представлен поиск лучших гиперпараметров и модели наивного Байеса

```
naive_bayes_models = {
    'GaussianNB': GaussianNB(),
    #'MultinomialNB': MultinomialNB(),
    'BernoulliNB': BernoulliNB(),
}

param_grid = {
    'GaussianNB': {},
    #'MultinomialNB': {'alpha': [0.1, 0.5, 1.0]},
    'BernoulliNB': {'alpha': [0.1, 0.5, 1.0], 'binarize': [0.0, 0.1, 0.2]},
}
```

```
# Выбор метрики для оценки моделей
scoring_metric = 'accuracy'

# Обучение и оценка моделей с использованием GridSearchCV
best_models = {}
for model_name, model in naive_bayes_models.items():
    grid_search = GridSearchCV(model, param_grid[model_name], scoring=scoring_metric, cv=5, n_jobs=-1)
    grid_search.fit(x_train, y_train)

    best_models[model_name] = grid_search.best_estimator_
# Оценка наилучшей модели на тестовом наборе
best_model_name = max(best_models, key=lambda k: grid_search.cv_results_['mean_test_score'][grid_search.best_index_])
best_model = best_models[best_model_name]


# Вывод результатов
print(f"Лучшая модель: {best_model_name}")
print(f"Лучшие параметры: {grid_search.best_params_}")
```

```
Лучшая модель: GaussianNB
Лучшие параметры: {'alpha': 1.0, 'binarize': 0.2}
```

Рисунок 15 – Перекрестная проверка гиперпараметров

```
    best_model.fit(x_train, y_train)

    y_pred = best_model.predict(x_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Точность модели : {accuracy}')

    print(classification_report(y_test, y_pred))
```

```
Точность модели : 0.6035696741444244
              precision    recall  f1-score   support

           0       0.54      0.69      0.61      5374
           1       0.69      0.53      0.60      6840

    accuracy                           0.60     12214
   macro avg       0.61      0.61      0.60     12214
weighted avg       0.62      0.60      0.60     12214
```

Рисунок 16 – Оценка точности модели

4.3 Преимущества и ограничения

Преимущества метода наивного Байеса:

- Эффективен и быстр в обучении, особенно при работе с большими наборами данных.
- Хорошо работает в условиях небольшой обучающей выборки.
- Прост в реализации и понимании.

Однако у метода наивного Байеса есть и ограничения:

- Предполагает независимость признаков, что может быть не всегда верным в реальных данных.
- Может давать неоптимальные результаты в случае, когда зависимости между признаками сильны.

5. Методы решающего дерева и случайного леса

5.1 Описание метода

Решающее дерево представляет собой структуру, состоящую из узлов и листьев, которая используется для принятия решений на основе вопросов о значениях признаков. Оно строит дерево, разбивая данные на подмножества на основе определенных признаков.

Принцип работы решающего дерева заключается в выборе наилучшего признака для разделения данных на каждом узле дерева. Этот процесс повторяется, пока не будет достигнуто условие остановки, такое как максимальная глубина дерева или минимальное количество объектов в листе.

5.2 Применение на модельных данных

-

На рисунке 17 представлен метод решающего дерева

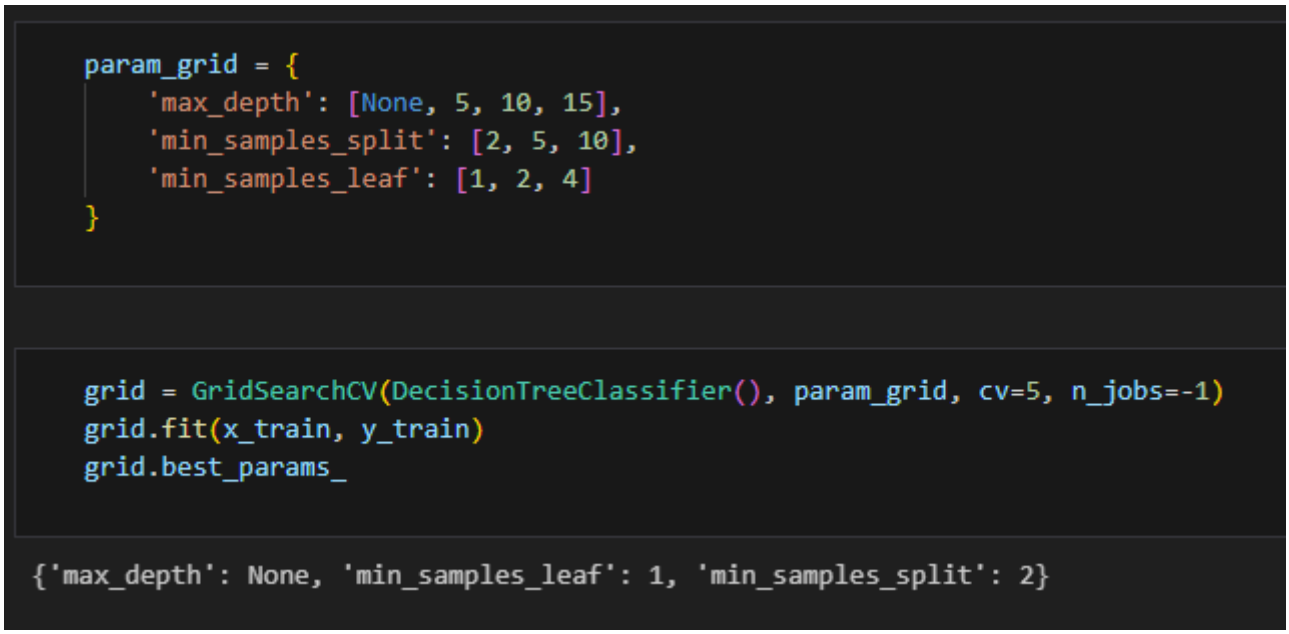


Рисунок 17 – Метод решающего дерева

На рисунке 18 представлены обучение решающего дерева и вывод результата

```
    y_pred = grid.best_estimator_.predict(x_test)

    print(f'Точность для решающего дерева : {accuracy_score(y_test, y_pred)}')
    print(classification_report(y_test, y_pred))
```

```
Точность для решающего дерева : 0.9968888161126576
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5374
           1       1.00      1.00      1.00      6840

    accuracy                           1.00     12214
   macro avg       1.00      1.00      1.00     12214
weighted avg       1.00      1.00      1.00     12214
```

Рисунок 18 – Обучение и вывод результата

На рисунке 19 представлен метод случайного дерева

```
param_grid = {
    'n_estimators': range(2, 10),
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2, 4]
}
```

```
grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, n_jobs=-1)
grid.fit(x_train, y_train)
grid.best_params_
```

```
{'max_depth': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 9}
```

Рисунок 19 Метод случайного дерева

На рисунке 20 представлены обучение случайного дерева и вывод результата.

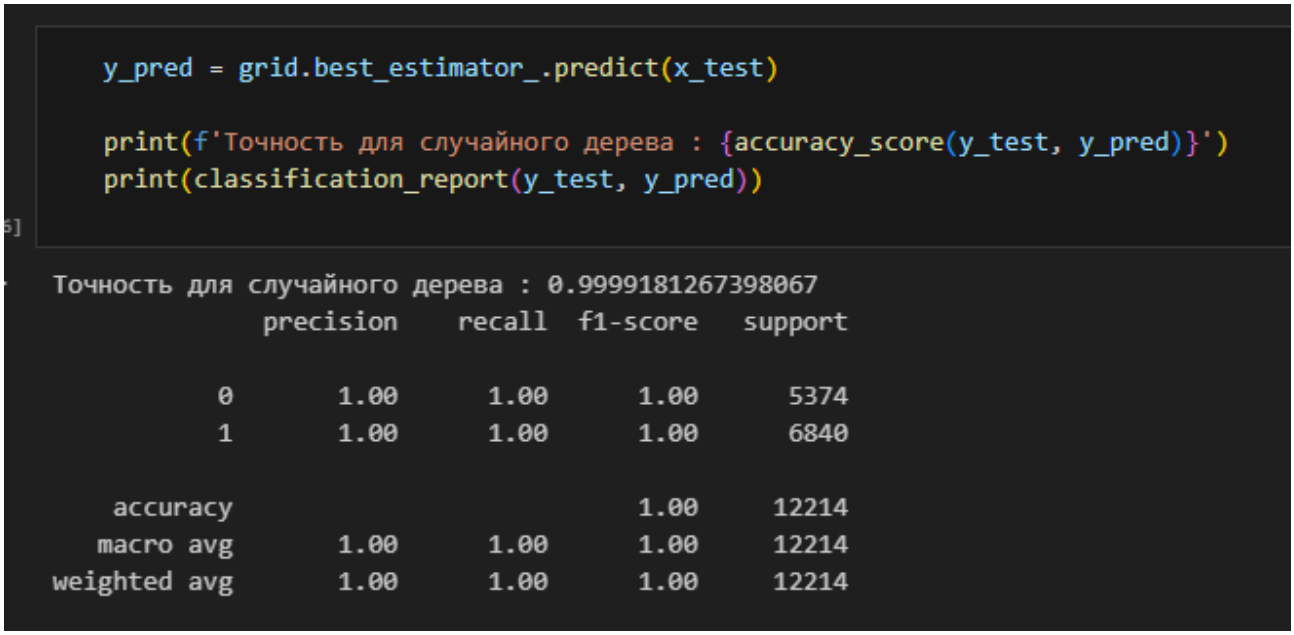


Рисунок 20 – Обучение и вывод результата

5.3 Сравнение с другими методами классификации

Решающее дерево и случайный лес являются эффективными методами классификации, которые обладают рядом преимуществ:

- Способны обрабатывать как категориальные, так и числовые данные.
- Позволяют визуализировать принятие решений.
- Могут обрабатывать большие объемы данных.

Однако у этих методов также есть некоторые ограничения:

- Склонны к переобучению на обучающих данных.
- Могут не улавливать сложные нелинейные зависимости.

По сравнению с линейными моделями, решающие деревья и случайный лес могут быть более гибкими в обработке сложных данных, но требуют осторожной настройки параметров для предотвращения переобучения. Их эффективность часто зависит от природы данных и особенностей задачи классификации.

6. Метод CatBoost

6.1 Описание метода

CatBoost (Categorical Boosting) – это высокоэффективная библиотека градиентного бустинга, специально разработанная для работы с категориальными признаками. Она представляет собой мощный алгоритм машинного обучения, который широко применяется в задачах классификации, регрессии и ранжирования.

6.2 Применение на модельных данных

На рисунке 21 представлена реализация метода CatBoost

```
param_grid = {
    'depth': [1, 4, 7, 10],
    'learning_rate': [0.01, 0.1, 1],
    'l2_leaf_reg': [1, 3, 5, 9],
    'iterations': [100, 200,],
    'depth': [0, 3, 6],
    'loss_function': ['MultiClass', 'Logloss']
}
```

```
grid = GridSearchCV(CatBoostClassifier(), param_grid, cv=5, n_jobs=-1)
grid.fit(x_train, y_train)
grid.best_params_
```

Рисунок 21 – Метод CatBoost

На рисунке 22 преставлен вывод результата работы метода CatBoost

```
model = CatBoostClassifier(**grid.best_params_, verbose=False)
model.fit(x_train, y_train)

y_pred = model.predict(x_test)

print(f"Точность модели: {accuracy_score(y_test, y_pred)}")
print(classification_report(y_test, y_pred))
```

```
Точность модели: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5374
           1       1.00      1.00      1.00      6840

    accuracy                           1.00     12214
   macro avg       1.00      1.00      1.00     12214
weighted avg       1.00      1.00      1.00     12214
```

Рисунок 22 – Вывод результата

CatBoost обладает мощными возможностями для работы с категориальными данными и является одним из популярных алгоритмов для задач машинного обучения, особенно в случаях, когда требуется обработка сложных и разнородных данных без предварительной подготовки.

Заключение

В рамках данного исследования были рассмотрены и проанализированы различные методы машинного обучения, каждый из которых представляет собой мощный инструмент для решения задач классификации и регрессии.

Выбор конкретного метода зависит от природы данных, размера выборки, задачи и требований к точности предсказаний. Каждый метод имеет свои преимущества и ограничения, и правильный выбор может существенно повлиять на результаты анализа. Для повышения эффективности и точности модели часто требуется комбинирование нескольких методов или тщательная настройка параметров.

В целом, эта работа служит важным введением в мир машинного обучения и предоставляет базовые знания о различных методах, что поможет в выборе и применении наиболее подходящего метода для конкретной задачи.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Основы машинного обучения, лекция 2 — метод k ближайших соседей: https://www.youtube.com/watch?v=X081VuXB1og&list=PLEwK9wdS5g0oCRxBzxsq9lkJkzMgzWiyg&index=2
2. Основы машинного обучения, лекция 4 — линейная регрессия: https://www.youtube.com/watch?v=8RAXDT_5_js&list=PLEwK9wdS5g0oCRxBzxsq9lkJkzMgzWiyg&index=24
3. Андрей Бурков. Машинное обучение без лишних слов