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»

**РЕШЕНИЕ ЗАДАЧИКЛАССИФИКАЦИИ НА ПРИМЕРЕ АПЕЛЬСИНОВ И ГРЕПФРУТОВ**

Разработчики проекта:

Дьякова Ирина Андреевна,

Мальцев Владислав Леонидович

Пермь, 2024

ОГЛАВЛЕНИЕ

[ПАСПОРТ ПРОЕКТА 3](#_Toc185807349)

[СОДЕРЖАНИЕ ПРОЕКТА 4](#_Toc185807350)

[Анализ предметной области 4](#_Toc185807351)

[Исходные данные 5](#_Toc185807352)

[Реализация проекта 6](#_Toc185807353)

[Этап 1. Подготовка данных к анализу 6](#_Toc185807354)

[Этап 2. Предварительный анализ данных 8](#_Toc185807355)

[Этап 3. Корреляционный анализ данных 13](#_Toc185807356)

[Этап 4. Моделирование 15](#_Toc185807357)

[Этап 5. Визуализация решений 16](#_Toc185807358)

[Заключение 21](#_Toc185807359)

[Список использованных источников и литературы 22](#_Toc185807360)

[Приложения 23](#_Toc185807361)

ПАСПОРТ ПРОЕКТА

Название проекта: Решение задачи классификации на примере апельсинов и грейпфрутов.

Сведения об авторах: Мальцев Владислав Леонидович, Дьякова Ирина Андреевна.

Цель: проанализировать данные и построить модель бинарной классификации для апельсинов и грейпфрутов.

Задачи:

1. Исследовать проблему и обосновать значимость темы.
2. Загрузить данные и подготовить их для анализа.
3. Осуществить предварительный анализ данных: обработка выбросов, проверкаа нормальности распределения данных и корреляционный анализ.
4. Выполнить моделирование зависимости целевого признака от факторных, подобрать оптимальную модель, оценить ее качество и выполнить прогнозирование.
5. Интерпретировать полученные результаты и сделать выводы.

Краткое описание проекта:

Необходимо проанализировать данные о фруктах (апельсинах и грейпфрутах) и определить, можно ли описать зависимость типа фрукта от имеющихся факторных переменных. Интерпретировать полученные результаты и сделать выводы.

Конкретные ожидаемые результаты:

Разработанная классификационная модель.

СОДЕРЖАНИЕ ПРОЕКТА

Анализ предметной области

Разделение апельсинов и грейпфрутов — задача, которая кажется на первый взгляд довольно простой и очевидной для человека. Однако при внимательном наблюдении становится ясно, что ошибки в классификации все же могут возникать, даже при наличии навыков и опыта. Такие ошибки могут быть вызваны разнообразием форм, размеров и цветов этих цитрусовых фруктов, а также наличием различных сортов, которые могут значительно отличаться друг от друга. Таким образом, важно учитывать не только визуальные характеристики, но и другие параметры, такие как вес и диаметр, чтобы добиться более точной классификации.

Актуальность этой задачи в наше время трудно переоценить. В условиях постоянного роста населения и увеличения спроса на свежие фрукты необходимо оптимизировать процессы сортировки и продажи товара. Современные технологии, такие как машинное обучение и анализ больших данных, могут существенно повысить точность и скорость классификации, позволяя выделять ценную информацию, которая может быть использована для управления качеством продукции, оптимизации логистики и повышения уровня обслуживания клиентов.

Таким образом, создание точного алгоритма, который способен различать апельсины и грейпфруты, не только актуально, но и представляет собой важный шаг на пути к более современным и эффективным методам работы в агропромышленном комплексе. Это, в конечном итоге, приведет к повышению прибыльности и устойчивости бизнеса в долгосрочной перспективе.

Цель: проанализировать данные по фруктам и построить модель бинарной классификации для апельсинов и грейпфрутов.

Задачи:

1. Исследовать проблему и обосновать значимость темы.
2. Загрузить данные и подготовить их для анализа.
3. Осуществить предварительный анализ данных: обработка выбросов, проверкаа нормальности распределения данных и корреляционный анализ.
4. Выполнить моделирование зависимости целевого признака от факторных, подобрать оптимальную модель, оценить ее качество и выполнить прогнозирование.
5. Интерпретировать полученные результаты и сделать выводы.

Исходные данные

В настоящей работе анализируется набор данных, в котором содержится информация об апельсинах и грейпфрутах. Он включает в себя параметры, такие как цвет, вес и диаметр «средних» образцов фруктов. Это позволяет создать более обширный набор, который учитывает широкий диапазон значений, что в свою очередь благоприятно сказывается на точности классификации.

Список колонок анализируемого набора данных:

1. name – название фрукта (целевая переменная).
2. diameter – диаметр.
3. weight – вес.
4. red – яркость красной компоненты в модели RGB.
5. green – яркость зелёной компоненты в модели RGB.
6. blue – яркость синей компоненты в модели RGB.

Последние три компоненты задают цвет фрукта в модели RGB.

Выдвинем гипотезу исследования: вид фрукта (апельсин или грейпфрут) может быть предсказанным по двум показателям, а именно веса и яркости зеленой компоненты в модели RGB.

Реализация проекта

1. Подготовка данных к анализу

Подключаем необходимые библиотеки, которые помогут эффективно работать с данными:

from google.colab import drive

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats

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

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import accuracy\_score

from mlxtend.plotting import plot\_decision\_regions

Загружаем данные, которые храниться в облачного хранилища:

drive.mount('/content/drive')

dataset = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/ВКР/citrus.csv')

dataset.head(5)



Рисунок 1. Исходный датафрейм

Важно убедится, что все количественные столбцы имеют числовой тип. Это является необходимым для корректного выполнения математических операций и построения моделей.

dataset.info()

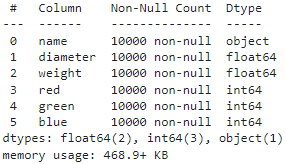


Рисунок 2. Типы данных

Видим, что все данные кроме целевой переменной имеют числовой тип.

Следующий этап: проверка данных на наличие пропусков. Они могут негативно повлиять на качество анализа и моделирования. Поэтому их важно выявить и устранить. Посмотрим количество пропусков по колонкам:

dataset.isnull().sum()

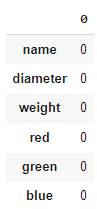


Рисунок 3. Наличие пропусков

Результаты проверки показали, что пропусков в колонках нет, что говорит о высоком качестве данных.

Далее необходимо перейти к анализу целевой переменной. Целевая переменная – это ключевой элемент анализа, определяющий класс каждого объекта. Важно понимать, какие значения принимает целевая переменная:

dataset['name'].unique()

IMG_256

Рисунок 4. Значения целевой переменной

Также необходимо убедиться, что набор данных сбалансирован. Это имеет важное значение. Если один класс будет значительно преобладать над другим, то модель может предвзято относиться в пользу большего класса.

dataset['name'].value\_counts()

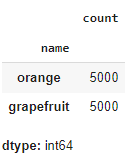


Рисунок 5. Сбалансированность классов

После выполнения всех шагов, которые были описаны выше, был получен набор данных, который полностью готов к анализу.

1. Предварительный анализ данных

Чтобы лучше понять структуру и характеристики данных, необходимо провести детальный анализ. Первым необходимо провести вычисления описательных статистик для каждой колонки с числовым типом. Они включают в себя среднее арифметическое, моду, медиану, дисперсию и стандартное отклонение.

dataset.describe()

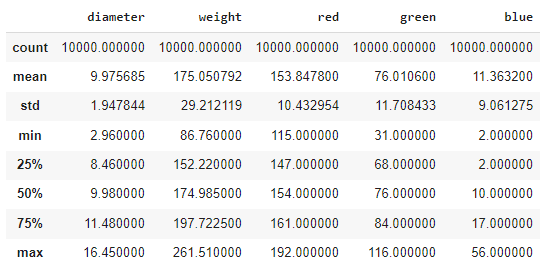


Рисунок 6. Описательные статистики

Далее необходимо проверить данные на наличие выбросов, то есть на значения, которые значительно отличаются. Для их выявления используем boxplot для каждой числовой колнки.

plt.figure(figsize=(18, 10))

for idx, column in enumerate(dataset.columns[1:]):

plt.subplot(2, 3, idx + 1)

sns.boxplot(x=dataset[column])

plt.title(f'Boxplot of {column}')

plt.tight\_layout()

plt.show()

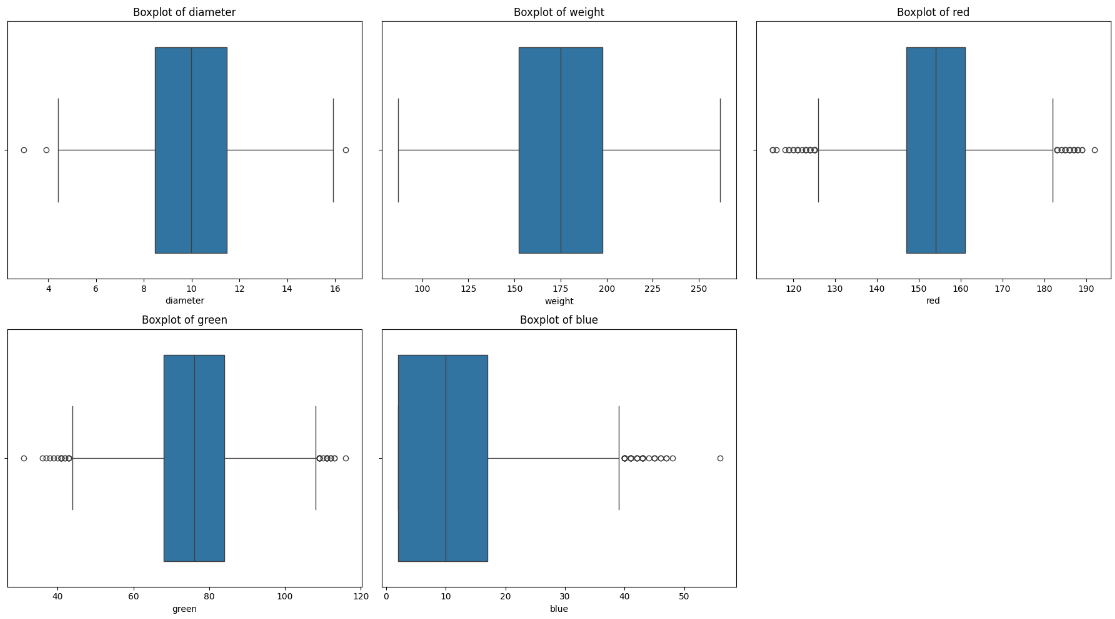


Рисунок 7. Boxplot с выбросами

Заметно, что есть явные выбросы, кроме столбца weight. Попробуем их удалить.

def remove\_outliers(dataframe, feature, condition\_feature, \_name):

feature\_fraud = dataframe[feature].loc[dataframe[condition\_feature] == \_name].values

Q1, Q3 = np.percentile(feature\_fraud, [25, 75])

IQR = Q3 - Q1

low\_boundary, high\_boundary = Q1 - 1.5 \* IQR, Q3 + 1.5 \* IQR

outliers = [x for x in feature\_fraud if x < low\_boundary or x > high\_boundary]

dataframe\_without\_outliers = dataframe.drop(dataframe[(dataframe[feature] > high\_boundary) | (dataframe[feature] < low\_boundary)].index)

return dataframe\_without\_outliers, len(outliers)

features = ['diameter', 'red', 'green', 'blue']

for feature in features:

new\_df\_1, outlier\_count = remove\_outliers(dataset, feature, 'name', 'grapefruit')

print('Количество выбросов: {}'.format(outlier\_count), feature)

new\_df\_2, outlier\_count = remove\_outliers(new\_df\_1, feature, 'name', 'orange')

print('Количество выбросов: {}'.format(outlier\_count), feature, '\n')

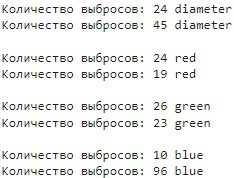


Рисунок 8. Количество удаленных выбросов

Посмотрим на диаграммы boxplot после удаления выбросов:

plt.figure(figsize=(18, 10))

for idx, column in enumerate(new\_df\_2.columns[1:]):

plt.subplot(2, 3, idx + 1)

sns.boxplot(x=new\_df\_2[column])

plt.title(f'Boxplot of {column}')

plt.tight\_layout()

plt.show()

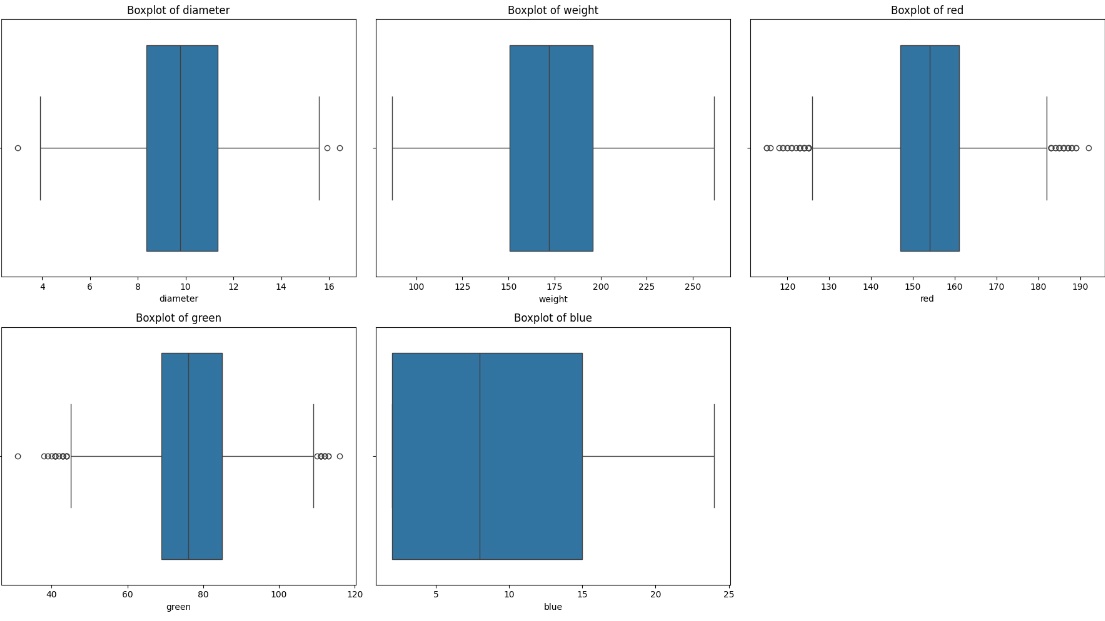


Рисунок 9. Boxplot после удаления выбросов

От наиболее заметных выбросов удалось избавиться. Так же можно сказать, что они полностью пропали у столбца blue.

Также проверим данные на нормальность распределения двумя способами:

1. Построим гистограмму и сделаем предположение.
2. Выполним статистический тест на нормальность и убедимся в правильности своих предположений.

Сначала построим гистограммы для всех числовых колонок:

plt.figure(figsize=(18, 10))

for idx, column in enumerate(new\_df\_2.columns[1:]):

plt.subplot(2, 3, idx + 1)

sns.histplot(dataset[column], kde=True)

plt.title(f'Histogram of {column}')

plt.tight\_layout()

plt.show()

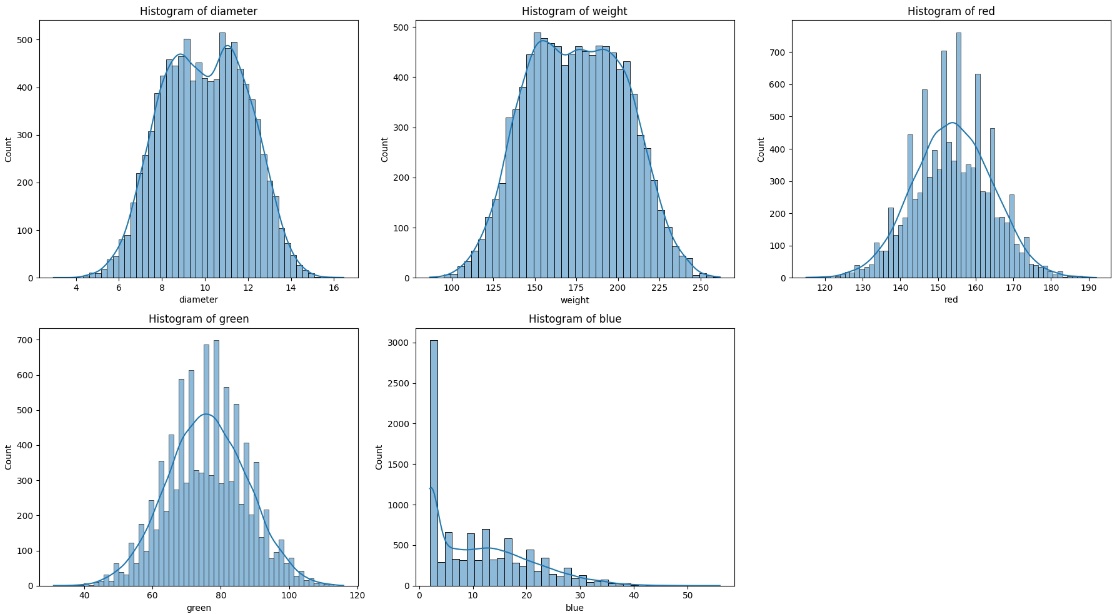


Рисунок 10. Гистограммы

Визуально видно, что данные имеют не нормальное распределение. Чтобы в этом убедиться используем тест Шапиро-Уилка. В качестве уровня значимости возьмем значение 0.05. Если значение получится меньше этого уровня, то наше предположение будет верно.

alpha = 0.05

for column in new\_df\_2.columns[1:]:

# Тест Шапиро-Уилка

shapiro\_test = stats.shapiro(new\_df\_2[column])

print(f'Tест Шапиро-Уилка: статистика = {shapiro\_test.statistic}, p-значение = {shapiro\_test.pvalue}')

# Проверка гипотезы

if shapiro\_test.pvalue > alpha:

print('Данные нормально распределены.')

else:

print('Данные не нормально распределены.')

print()

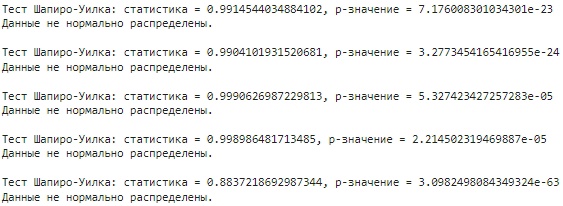


Рисунок 11. Тест Шапира-Уилка

На втором этапе был проведен более детальный анализ, что позволило удалить наиболее заметные выбросы. Также убедились в том, что данные во всех столбцах имеют не нормальное распределение. Этим мы смогли подготовить данные к дальнейшему анализу и моделированию.

1. Корреляционный анализ данных

Корреляционный анализ показывает степень тесноты статистической связи между анализируемыми переменными. Если бы данные имели нормальное распределение, то необходимо было бы использовать коэффициент корреляции Пирсона. А в случае с данными, которые имеют ненормальное распределение матрица корреляции будет строиться с помощью коэффициентов корреляции Спирмена.

Чтобы была возможность посмотреть зависимости и для целевой переменной, необходимо преобразовать столбец с name из категориального в числовую тип данных, где за апельсины будет отвечать значение 0, а 1 – грейпфруты.

name = {'orange': 0, 'grapefruit': 1}

new\_df\_2['name'] = new\_df\_2['name'].map(name)

Матрицу корреляции отобразим с помощью диаграммы «тепловая карта» (heatmap), используя ранговый коэффициент кореляции Спирмена.

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

correlation\_matrix = new\_df\_2.corr(method='spearman')

sns.heatmap(correlation\_matrix, annot=True, fmt='.2f', cmap='coolwarm', square=True, cbar\_kws={"shrink": .8})

plt.title('Корреляционная матрица')

plt.show()

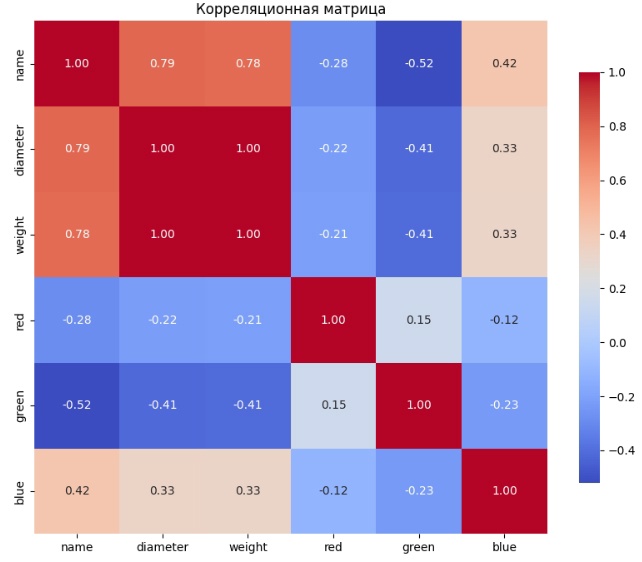


Рисунок 12. Матрица корреляции

По данным из матрицы корреляции видно, что переменные weight и diameter напрямую зависят друг от друга, поэтому в качестве факторных признаков столбец diameter использоваться не будет. Для обучения необходимо разделить данные на факторные признаки и целевую переменную.

X = new\_df\_2.drop(['name', 'diameter'], axis=1)

y = new\_df\_2['name']

Выполним нормализация числовых столбцов – факторных признаков путем приведения их значений к диапазону от 0 до 1.

scaler = MinMaxScaler()

X = pd.DataFrame(scaler.fit\_transform(X), columns=X.columns)

X.head()

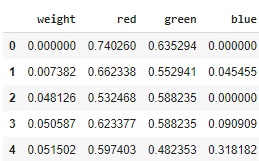


Рисунок 13. Нормализованные данные

На третьем этапе был проведен корреляционный анализ, который позволил выбрать факторные признаки для обучения модели. Данные для моделирования готовы.

1. Моделирование

На этапе моделирования первым и важным действием является разбиние даннх на обучающую и тестовую выборки в соотношении 8 к 2

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15, stratify=y)

Также необходимо выбрать модель для обучения, для этого создам несколько и посмотрим какая лучше подойдет. .

models = []

models.append(('LR', LogisticRegression(solver='lbfgs', max\_iter=1000)))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC(gamma='auto')))

Теперь оценим метрики каждой модели.

results = []

model\_names = []

for name, model in models:

kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)

cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

model\_names.append(name)

print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))

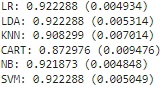


Рисунок 14. Метрики различных моделей

Результаты показали, что модели LR, LDA и SVM имеют одинаковую среднюю точность, но модель логистической регрессии имеет наименьшее стандартное отклонение, что указывает на её стабильность. Таким образом для обучения наших данных мы выберем модель логистической регрессии.

На обучающей выборке построим модель логистической регрессии, используя в качестве целевой и факторных переменных отобранные ранее столбцы. И сделаем предсказания на тестовых данных.

sk\_lr = LogisticRegression(max\_iter=10000)

sk\_lr.fit(X\_train, y\_train)

sk\_lr\_pred\_res = sk\_lr.predict(X\_test)

sk\_lr\_accuracy = accuracy\_score(y\_test, sk\_lr\_pred\_res)

print(f'sk LR accuracy: {sk\_lr\_accuracy}')



Рисунок 15. Предсказание на тестовой выборке

Этап моделирования позволил определить оптимальную модель для классификации апельсинов и грейпфрутов, а именно логистическую регрессию. Что показало очень хорошие результаты после обучение и проверки на тестовых данных.

1. Визуализация решений

После того, как обучение прошло успешно, можно лучше ознакомиться с результатами. Матрица ошибок наглядно покажет сколько из тестового набора данных было правильных ответов.

from sklearn.metrics import confusion\_matrix

conf\_matrix = confusion\_matrix(y\_test, sk\_lr\_pred\_res)

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Предсказанные')

plt.ylabel('Настоящие')

plt.title('Confusion Matrix')

plt.show()

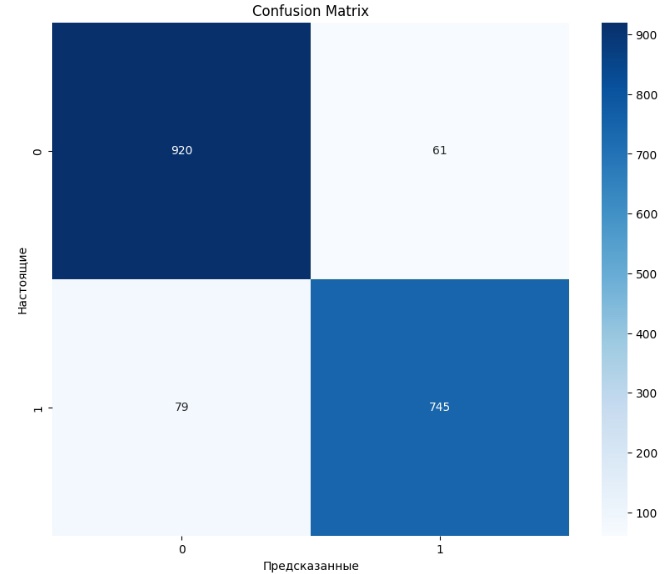


Рисунок 16. Матрица ошибок

Как видно из матрицы ошибок, наша модель совершает ошибки, но их меньше 10%.

Также вычислим другие показатели качества классификации:

1. Accuracy – для правильных предсказаний.
2. Precision – точность положительных предсказаний.
3. Recall – полнота, то есть на сколько хорошо модель находит положительные случаи.

from sklearn import model\_selection, preprocessing, feature\_selection, ensemble, linear\_model, metrics, decomposition

accuracy = metrics.accuracy\_score(y\_test, sk\_lr\_pred\_res)#Оценим точность классификации.

recall = metrics.recall\_score(y\_test, sk\_lr\_pred\_res, average="weighted")

precision = metrics.precision\_score(y\_test, sk\_lr\_pred\_res, average="weighted")

print("Recall (all 1s predicted right):", round(recall,2))

print("Precision (confidence when predicting a 1):", round(precision,2))

print("Detail:")

print(metrics.classification\_report(y\_test, sk\_lr\_pred\_res, target\_names=[str(i) for i in np.unique(y\_test)]))

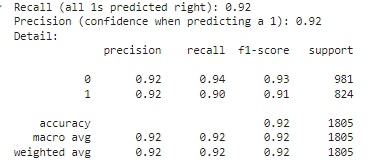


Рисунок 17. Показатели качества классификации

В начале была выдвинута гипотеза о том, что с помощью всего двух метрик: веса и яркости зелёной компоненты в модели RGB можно определять вид фрукта. Чтобы это проверить можно в качестве факторных признаков, взять эти две метрики и обучить модель.

Xs = new\_df\_2[['weight', 'green']]

ys = new\_df\_2["name"].values

sk\_lr2 = LogisticRegression(solver='lbfgs', max\_iter=10000, multi\_class='auto')

sk\_lr2.fit(Xs, ys)

X1\_lr\_train, X1\_lr\_test, y1\_train, y1\_test = train\_test\_split(Xs, ys, random\_state=0)

sk\_lr2.fit(X1\_lr\_train, y1\_train)

Для лучшего понимания того, как модель классифицирует данные, можно визуализировать границы решений.

plt.figure()

plt.title('Logistic Regression surface with transformed features')

plot\_decision\_regions(X=Xs.values, y=ys, clf=sk\_lr2)

plt.show()

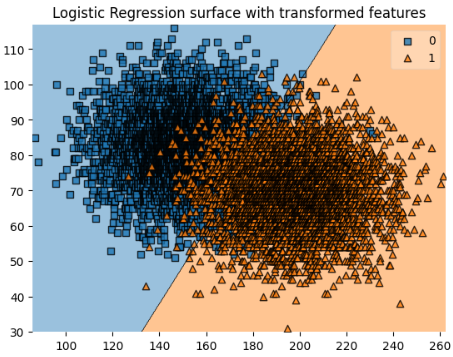


Рисунок 18. Визуализация границ решений

После визуализации границ решений, также можно посмотреть показатели качества классификации, что поможет лучше понять об истинности или ложности гипотезы.

predicted = sk\_lr2.predict(X1\_lr\_test)

accuracy = metrics.accuracy\_score(y1\_test, predicted)#Оценим точность классификации.

recall = metrics.recall\_score(y1\_test, predicted, average="weighted")

precision = metrics.precision\_score(y1\_test, predicted, average="weighted")

print("Recall (all 1s predicted right):", round(recall,2))

print("Precision (confidence when predicting a 1):", round(precision,2))

print("Detail:")

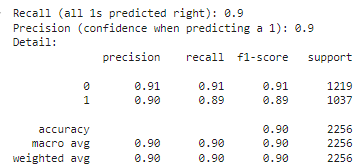


Рисунок 19. Показатели качества классификации по двум метрикам

Посмотрев на результаты, можно увидеть, что точность предсказания составляет около 90%. Что является довольно хорошим показателям и говорит об истинности гипотезы. А значит, зная вес фрукта и яркость зеленой компоненты, можно определить его вид: апельсин или грейпфрут.

Заключение

Модель классификации демонстрирует высокую производительность по всем ключевым метрикам. Высокая точность, полнота и F1-score указывают на то, что модель хорошо справляется с предсказанием как грейпфрутов, так и апельсинов. Это свидетельствует о том, что модель надежна и может быть использована для задач классификации с высокой уверенностью в её предсказаниях.

Таким образом, можно сделать вывод, что модель хорошо обучена и готова к применению в реальных условиях для задач классификации.

Для достижения цели были решены следующие задачи:

1. Исследовать проблему и обосновать значимость темы.
2. Загрузить данные и подготовить их для анализа.
3. Осуществить предварительный анализ данных: обработка выбросов, проверкаа нормальности распределения данных и корреляционный анализ.
4. Выполнить моделирование зависимости целевого признака от факторных, подобрать оптимальную модель, оценить ее качество и выполнить прогнозирование.
5. Интерпретировать полученные результаты и сделать выводы.

Построенная классификационная модель описывает различие апельсинов и грейпфрутов и может использоваться в торговле, производстве и бизнесе.

Гипотеза была подтверждена. Для определения фруктов достаточно использовать только две факторные переменные: вес и яркость зелёной компоненты в модели RGB. Хоть и модель показывает чуть хуже результаты, чем при использовании остальных факторных переменных, возможно можно добиться более лучшего результата если значений в выборке будет ещё больше.

Список использованных источников и литературы

1. Как работает алгоритм k-means [Электронный ресурс] URL: [https://habr.com/ru/articles/527334/](https://vk.com/away.php?to=https%3A%2F%2Fhabr.com%2Fru%2Farticles%2F527334%2F&utf=1) (дата обращения 22.12.2024).
2. Какие существуют методы определения выбросов в данных
3. [Электронный ресурс] URL: <https://sky.pro/media/kakiesushhestvuyut-metody-opredeleniya-vybrosov-v-dannyh/> (дата обращения 22.12.2024).
4. Логистическая регрессия в машинном обучении с Python [Электронный ресурс] URL: [https://datafinder.ru/products/logisticheskaya-regressiya-v-mashinnom-obuchenii-s-python](https://vk.com/away.php?to=https%3A%2F%2Fdatafinder.ru%2Fproducts%2Flogisticheskaya-regressiya-v-mashinnom-obuchenii-s-python&utf=1) (дата обращения 22.12.2024).
5. sklearn.metrics.accuracy\_score [Электронный ресурс] URL: [https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.accuracy\_score.html](https://vk.com/away.php?to=https%3A%2F%2Fscikit-learn.org%2F1.5%2Fmodules%2Fgenerated%2Fsklearn.metrics.accuracy_score.html&utf=1) (дата обращения 22.12.2024).
6. Confusion Matrix in Machine Learning [Электронный ресурс] URL: [https://www.geeksforgeeks.org/confusion-matrix-machine-learning/](https://vk.com/away.php?to=https%3A%2F%2Fwww.geeksforgeeks.org%2Fconfusion-matrix-machine-learning%2F&utf=1) (дата обращения 22.12.2024).

Приложения

Приложение 1

Программный код

from google.colab import drive

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

from scipy import stats

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

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.model\_selection import StratifiedKFold

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import accuracy\_score

from mlxtend.plotting import plot\_decision\_regions

drive.mount('/content/drive')

dataset = pd.read\_csv('/content/drive/MyDrive/Colab Notebooks/ВКР/citrus.csv')

dataset.head(10)

dataset.info()

dataset.isnull().sum()

dataset['name'].unique()

dataset['name'].value\_counts()

dataset.describe()

plt.figure(figsize=(18, 10))

for idx, column in enumerate(dataset.columns[1:]):

plt.subplot(2, 3, idx + 1)

sns.boxplot(x=dataset[column])

plt.title(f'Boxplot of {column}')

plt.tight\_layout()

plt.show()

def remove\_outliers(dataframe, feature, condition\_feature, \_name):

feature\_fraud = dataframe[feature].loc[dataframe[condition\_feature] == \_name].values

Q1, Q3 = np.percentile(feature\_fraud, [25, 75])

IQR = Q3 - Q1

low\_boundary, high\_boundary = Q1 - 1.5 \* IQR, Q3 + 1.5 \* IQR

outliers = [x for x in feature\_fraud if x < low\_boundary or x > high\_boundary]

dataframe\_without\_outliers = dataframe.drop(dataframe[(dataframe[feature] > high\_boundary) | (dataframe[feature] < low\_boundary)].index)

return dataframe\_without\_outliers, len(outliers)

features = ['diameter', 'red', 'green', 'blue']

for feature in features:

new\_df\_1, outlier\_count = remove\_outliers(dataset, feature, 'name', 'grapefruit')

print('Количество выбросов: {}'.format(outlier\_count), feature)

new\_df\_2, outlier\_count = remove\_outliers(new\_df\_1, feature, 'name', 'orange')

print('Количество выбросов: {}'.format(outlier\_count), feature, '\n')

f, ax = plt.subplots(4, 3, figsize=(15,30))

colors = ['#B3F9C5', '#f9c5b3']

i = 0

for col in features:

if i >= 4:

break

sns.boxplot(y=dataset[col], ax=ax[i, 0])

ax[i, 0].set\_title(f'Boxplot {col} до удаления выбросов')

sns.boxplot(y=new\_df\_1[col], ax=ax[i, 1])

ax[i, 1].set\_title(f'Boxplot {col} после удаления выбросов')

sns.boxplot(y=new\_df\_2[col], ax=ax[i, 2])

ax[i, 2].set\_title(f'Boxplot {col} после удаления выбросов')

i += 1

plt.tight\_layout()

plt.show()

plt.figure(figsize=(18, 10))

for idx, column in enumerate(new\_df\_2.columns[1:]):

plt.subplot(2, 3, idx + 1)

sns.boxplot(x=new\_df\_2[column])

plt.title(f'Boxplot of {column}')

plt.tight\_layout()

plt.show()

plt.figure(figsize=(18, 10))

for idx, column in enumerate(new\_df\_2.columns[1:]):

plt.subplot(2, 3, idx + 1)

sns.histplot(dataset[column], kde=True)

plt.title(f'Histogram of {column}')

plt.tight\_layout()

plt.show()

alpha = 0.05

for column in new\_df\_2.columns[1:]:

shapiro\_test = stats.shapiro(new\_df\_2[column])

print(f'Tест Шапиро-Уилка: статистика = {shapiro\_test.statistic}, p-значение = {shapiro\_test.pvalue}')

if shapiro\_test.pvalue > alpha:

print('Данные нормально распределены.')

else:

print('Данные не нормально распределены.')

print()

name = {'orange': 0, 'grapefruit': 1}

new\_df\_2['name'] = new\_df\_2['name'].map(name)

new\_df\_2.head()

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

correlation\_matrix = new\_df\_2.corr(method='spearman')

sns.heatmap(correlation\_matrix, annot=True, fmt='.2f', cmap='coolwarm', square=True, cbar\_kws={"shrink": .8})

plt.title('Корреляционная матрица')

plt.show()

X = new\_df\_2.drop(['name', 'diameter'], axis=1)

y = new\_df\_2['name']

print(f'X shape: {X.shape} | y shape: {y.shape} ')

scaler = MinMaxScaler()

X = pd.DataFrame(scaler.fit\_transform(X), columns=X.columns)

X.head()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=15, stratify=y)

stats.ttest\_ind (a=y\_train, b=y\_test)

models = []

models.append(('LR', LogisticRegression(solver='lbfgs', max\_iter=1000)))

models.append(('LDA', LinearDiscriminantAnalysis()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('CART', DecisionTreeClassifier()))

models.append(('NB', GaussianNB()))

models.append(('SVM', SVC(gamma='auto')))

results = []

model\_names = []

for name, model in models:

kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)

cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')

results.append(cv\_results)

model\_names.append(name)

print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))

sk\_lr = LogisticRegression(max\_iter=10000)

sk\_lr.fit(X\_train, y\_train)

sk\_lr\_pred\_res = sk\_lr.predict(X\_test)

sk\_lr\_accuracy = accuracy\_score(y\_test, sk\_lr\_pred\_res)

print(f'sk LR accuracy: {sk\_lr\_accuracy}')

print(f'sk LR prediction: {sk\_lr\_pred\_res}')

from sklearn.metrics import confusion\_matrix

conf\_matrix = confusion\_matrix(y\_test, sk\_lr\_pred\_res)

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

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.xlabel('Предсказанные')

plt.ylabel('Настоящие')

plt.title('Confusion Matrix')

plt.show()

from sklearn import model\_selection, preprocessing, feature\_selection, ensemble, linear\_model, metrics, decomposition

accuracy = metrics.accuracy\_score(y\_test, sk\_lr\_pred\_res)

recall = metrics.recall\_score(y\_test, sk\_lr\_pred\_res, average="weighted")

precision = metrics.precision\_score(y\_test, sk\_lr\_pred\_res, average="weighted")

print("Recall (all 1s predicted right):", round(recall,2))

print("Precision (confidence when predicting a 1):", round(precision,2))

print("Detail:")

print(metrics.classification\_report(y\_test, sk\_lr\_pred\_res, target\_names=[str(i) for i in np.unique(y\_test)]))

y\_s = y\_train

X\_2d= X\_train[['weight',  'green']].values

y\_2d = y\_s.values

sk\_lr1 = LogisticRegression(solver='lbfgs', max\_iter=10000, multi\_class='auto')

sk\_lr1.fit(X\_2d, y\_2d)

plt.title('LR surface with original features')

plot\_decision\_regions(X=X\_2d, y=y\_2d, clf=sk\_lr1)

Xs = new\_df\_2[['weight',  'green']]

ys = new\_df\_2["name"].values

sk\_lr2 = LogisticRegression(solver='lbfgs', max\_iter=10000, multi\_class='auto')

sk\_lr2.fit(Xs, ys)

X1\_lr\_train, X1\_lr\_test, y1\_train, y1\_test = train\_test\_split(Xs, ys, random\_state=0)

sk\_lr2.fit(X1\_lr\_train, y1\_train)

plt.figure()

plt.title('Logistic Regression surface with transformed features')

plot\_decision\_regions(X=Xs.values, y=ys, clf=sk\_lr2)

plt.show()

predicted = sk\_lr2.predict(X1\_lr\_test)

accuracy = metrics.accuracy\_score(y1\_test, predicted)

recall = metrics.recall\_score(y1\_test, predicted, average="weighted")

precision = metrics.precision\_score(y1\_test, predicted, average="weighted")

print("Recall (all 1s predicted right):", round(recall,2))

print("Precision (confidence when predicting a 1):", round(precision,2))

print("Detail:")

print(metrics.classification\_report(y1\_test, predicted, target\_names=[str(i) for i in np.unique(y\_test)]))