**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №4

«Линейные модели, SVM и деревья решений»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-62Б |  | преподаватель каф. ИУ5 |
| Щепетов Дмитрий |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

2024 г.

**Описание задания**

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.
4. Обучите следующие модели:
   * одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
   * SVM;
   * дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
6. Постройте график, показывающий важность признаков в дереве решений.
7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

**Текст программы**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

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

import matplotlib.ticker as ticker

from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier

import math

from enum import Enum

# from plotly.express import line

#import df

#import plotly.express as px

# %matplotlib inline

sns.set(style="ticks")

plt.style.use('ggplot')

pd.set\_option('display.max\_columns',24)

pd.set\_option('display.max\_colwidth',None)

pd.set\_option('display.float\_format', lambda x: '%.4f' % x)

encoding = 'windows-1250'

data=pd.read\_csv(r'C:\Users\Dima\Desktop\Homework\3 курс\ТМО\ЛР4\supply\_chain\_data.csv', encoding = encoding)

data\_train = pd.read\_csv(r'C:\Users\Dima\Desktop\Homework\3 курс\ТМО\ЛР4\supply\_chain\_data.csv', sep=",")

data.shape

data.info()

data.isnull().sum()

train\_data, test\_data = train\_test\_split(data, test\_size=0.2, random\_state=42)

import pandas as pd

import numpy as np

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

# Подготовка данных

# Предположим, что мы хотим использовать признаки 'Price', 'Availability', 'Stock levels' для предсказания 'Product Type'

X = data[['Price', 'Availability', 'Stock levels']]

y = data['Product type']

# Разделение данных на тренировочный и тестовый наборы

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Инициализация и обучение модели k-ближайших соседей

k = 3 # количество соседей

model = KNeighborsClassifier(n\_neighbors=k)

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

# Предсказание классов для тестового набора данных

y\_pred = model.predict(X\_test)

# Оценка точности модели

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

print("Accuracy:", accuracy)

# Создание переменных для обучающего и тестового наборов

train\_data = pd.concat([X\_train, y\_train], axis=1)

test\_data = pd.concat([X\_test, y\_test], axis=1)

# train\_data

# test\_data

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

ax = fig.add\_subplot(111, projection='3d')

# Обучающая выборка

ax.scatter(X\_train['Price'], X\_train['Availability'], X\_train['Stock levels'], c='b', marker='o', label='Train data')

# Тестовая выборка

ax.scatter(X\_test['Price'], X\_test['Availability'], X\_test['Stock levels'], c='r', marker='^', label='Test data')

ax.set\_xlabel('Price')

ax.set\_ylabel('Availability')

ax.set\_zlabel('Stock levels')

ax.set\_title('Supply Chain Analysis')

plt.legend()

plt.show()

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Подготовка данных

X = data[['Price', 'Availability', 'Stock levels']]

y = data['Product type']

# Разделение данных на тренировочный и тестовый наборы

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Инициализация и обучение модели логистической регрессии

model\_log = LogisticRegression()

model\_log.fit(X\_train, y\_train)

# Предсказание классов для тестового набора данных

y\_pred = model\_log.predict(X\_test)

# Оценка точности модели

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

print("Accuracy:", accuracy)

# Вывод коэффициентов модели

coefficients = model\_log.coef\_

intercept = model\_log.intercept\_

print("Coefficients:", coefficients)

print("Intercept:", intercept)

from sklearn.linear\_model import LinearRegression

# Подготовка данных

X = data[['Price', 'Availability', 'Stock levels']]

y = data['Revenue generated']

# Разделение данных на тренировочный и тестовый наборы

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Инициализация и обучение модели линейной регрессии

model\_linear = LinearRegression()

model\_linear.fit(X\_train, y\_train)

# Предсказание значений для тестового набора данных

y\_pred\_linear = model\_linear.predict(X\_test)

# Оценка качества модели (например, средняя абсолютная ошибка)

from sklearn.metrics import mean\_absolute\_error

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

print("Mean Absolute Error:", mae\_linear)

from sklearn.preprocessing import PolynomialFeatures

from sklearn.pipeline import make\_pipeline

from sklearn.linear\_model import LinearRegression

# Подготовка данных

X\_1 = data[['Price', 'Availability', 'Stock levels']]

y\_1 = data['Revenue generated']

# Создание полиномиальных признаков

degree = 2 # степень полинома

poly\_features = PolynomialFeatures(degree=degree)

X\_poly\_1 = poly\_features.fit\_transform(X\_1)

# Разделение данных на тренировочный и тестовый наборы

X\_1\_train, X\_1\_test, y\_1\_train, y\_1\_test = train\_test\_split(X\_poly\_1, y\_1, test\_size=0.3, random\_state=42)

# Инициализация и обучение модели полиномиальной регрессии

model\_poly = make\_pipeline(PolynomialFeatures(degree), LinearRegression())

model\_poly.fit(X\_1\_train, y\_1\_train)

# Предсказание значений для тестового набора данных

y\_pred\_poly = model\_poly.predict(X\_1\_test)

# Оценка качества модели (например, средняя абсолютная ошибка)

from sklearn.metrics import mean\_absolute\_error

mae\_poly = mean\_absolute\_error(y\_1\_test, y\_pred\_poly)

print("Mean Absolute Error (Polynomial Regression):", mae\_poly)

from sklearn.svm import SVC

# Подготовка данных

X = data[['Price', 'Availability', 'Stock levels']]

y = data['Product type']

# Разделение данных на тренировочный и тестовый наборы

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Инициализация и обучение модели SVM

model\_svm = SVC(kernel='linear') # Вы можете использовать различные ядра (например, 'linear', 'poly', 'rbf')

model\_svm.fit(X\_train, y\_train)

# Предсказание классов для тестового набора данных

y\_pred = model\_svm.predict(X\_test)

# Оценка точности модели

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

print("Accuracy:", accuracy)

# Вывод опорных векторов

support\_vectors = model\_svm.support\_vectors\_

print("Support vectors:", support\_vectors)

# Вывод коэффициентов для опорных векторов (веса)

coefficients = model\_svm.coef\_

print("Coefficients for support vectors:", coefficients)

from sklearn.tree import DecisionTreeClassifier

# Подготовка данных

X = data[['Price', 'Availability', 'Stock levels']]

y = data['Product type']

# Разделение данных на тренировочный и тестовый наборы

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Инициализация и обучение модели дерева решений

model\_dt = DecisionTreeClassifier()

model\_dt.fit(X\_train, y\_train)

# Предсказание классов для тестового набора данных

y\_pred = model\_dt.predict(X\_test)

# Оценка точности модели

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

print("Accuracy:", accuracy)

from sklearn.tree import plot\_tree

# Вывод важности признаков

feature\_importance = model\_dt.feature\_importances\_

print("Feature Importance:", feature\_importance)

# Вывод структуры дерева

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

plot\_tree(model\_dt, feature\_names=X.columns, class\_names=model\_dt.classes\_, filled=True)

plt.show()

from sklearn.metrics import confusion\_matrix, recall\_score, precision\_score, f1\_score

# Предсказания для модели логистической регрессии

y\_pred\_lr = model\_log.predict(X\_test)

# Предсказания для модели SVM

y\_pred\_svm = model\_svm.predict(X\_test)

# Предсказания для модели дерева решений

y\_pred\_dt = model\_dt.predict(X\_test)

# Оценка модели логистической регрессии

accuracy\_lr = accuracy\_score(y\_test, y\_pred\_lr)

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

recall\_lr = recall\_score(y\_test, y\_pred\_lr, average='weighted')

precision\_lr = precision\_score(y\_test, y\_pred\_lr, average='weighted')

f1\_lr = f1\_score(y\_test, y\_pred\_lr, average='weighted')

# Оценка модели SVM

accuracy\_svm = accuracy\_score(y\_test, y\_pred\_svm)

conf\_matrix\_svm = confusion\_matrix(y\_test, y\_pred\_svm)

recall\_svm = recall\_score(y\_test, y\_pred\_svm, average='weighted')

precision\_svm = precision\_score(y\_test, y\_pred\_svm, average='weighted')

f1\_svm = f1\_score(y\_test, y\_pred\_svm, average='weighted')

# Оценка модели дерева решений

accuracy\_dt = accuracy\_score(y\_test, y\_pred\_dt)

conf\_matrix\_dt = confusion\_matrix(y\_test, y\_pred\_dt)

recall\_dt = recall\_score(y\_test, y\_pred\_dt, average='weighted')

precision\_dt = precision\_score(y\_test, y\_pred\_dt, average='weighted')

f1\_dt = f1\_score(y\_test, y\_pred\_dt, average='weighted')

# Вывод результатов

print("Модель логистической регрессии:")

print("Accuracy:", accuracy\_lr)

print("Confusion Matrix:")

print(conf\_matrix\_lr)

print("Recall:", recall\_lr)

print("Precision:", precision\_lr)

print("F1 Score:", f1\_lr)

print()

print("Модель SVM:")

print("Accuracy:", accuracy\_svm)

print("Confusion Matrix:")

print(conf\_matrix\_svm)

print("Recall:", recall\_svm)

print("Precision:", precision\_svm)

print("F1 Score:", f1\_svm)

print()

print("Модель дерева решений:")

print("Accuracy:", accuracy\_dt)

print("Confusion Matrix:")

print(conf\_matrix\_dt)

print("Recall:", recall\_dt)

print("Precision:", precision\_dt)

print("F1 Score:", f1\_dt)

import matplotlib.pyplot as plt

# Получение важности признаков

feature\_importance = model\_dt.feature\_importances\_

# Названия признаков

feature\_names = X.columns

# Создание графика

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

plt.barh(feature\_names, feature\_importance)

plt.xlabel('Importance')

plt.ylabel('Feature')

plt.title('Feature Importance in Decision Tree Model')

plt.show()

plt.plot(feature\_names, feature\_importance, marker='o')

plt.xlabel('Feature')

plt.ylabel('Importance')

plt.title('Feature Importance in Decision Tree Model')

plt.xticks(rotation=90)

plt.show()

plt.scatter(feature\_names, feature\_importance)

plt.xlabel('Feature')

plt.ylabel('Importance')

plt.title('Feature Importance in Decision Tree Model')

plt.xticks(rotation=90)

plt.show()

plt.pie(feature\_importance, labels=feature\_names, autopct='%1.1f%%')

plt.title('Feature Importance in Decision Tree Model')

plt.show()

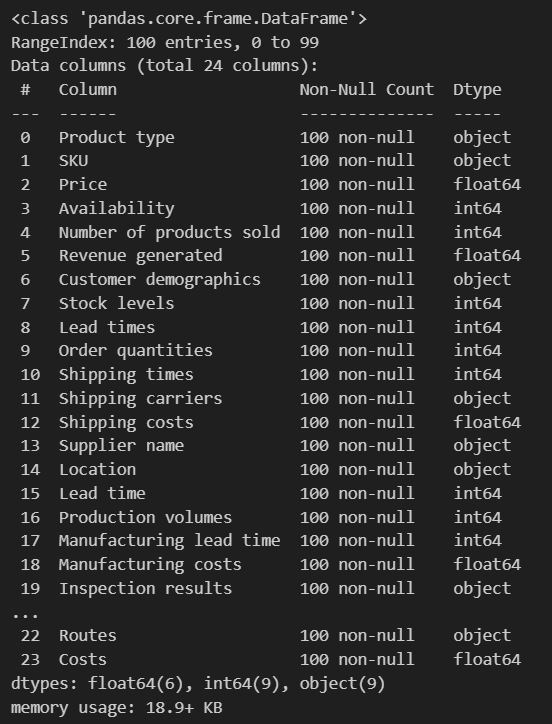
from sklearn.tree import export\_text

tree\_rules = export\_text(model\_dt, feature\_names=list(X.columns))

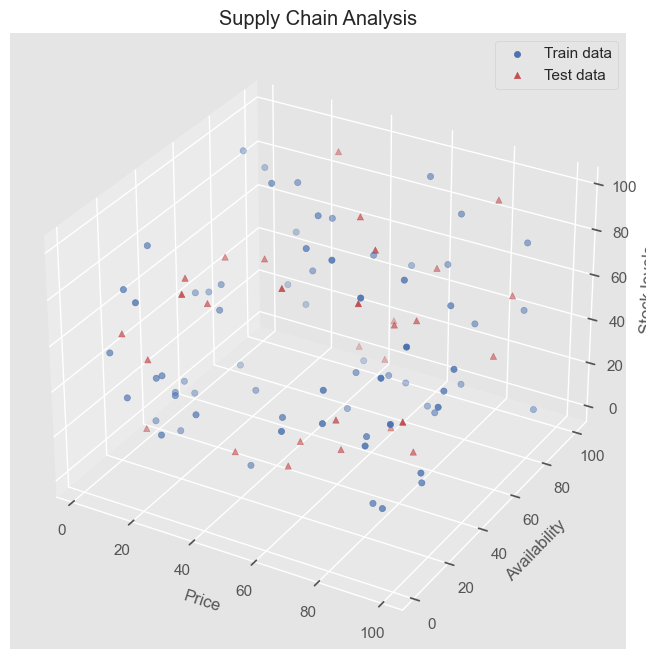
print(tree\_rules)

**Экранные формы с примерами выполнения программы**

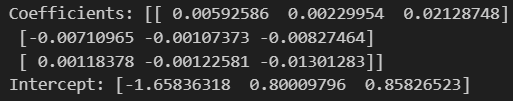
Просмотр информации о датасете



Разделение выборки на обучающую и тестовую



Логистическая регрессия



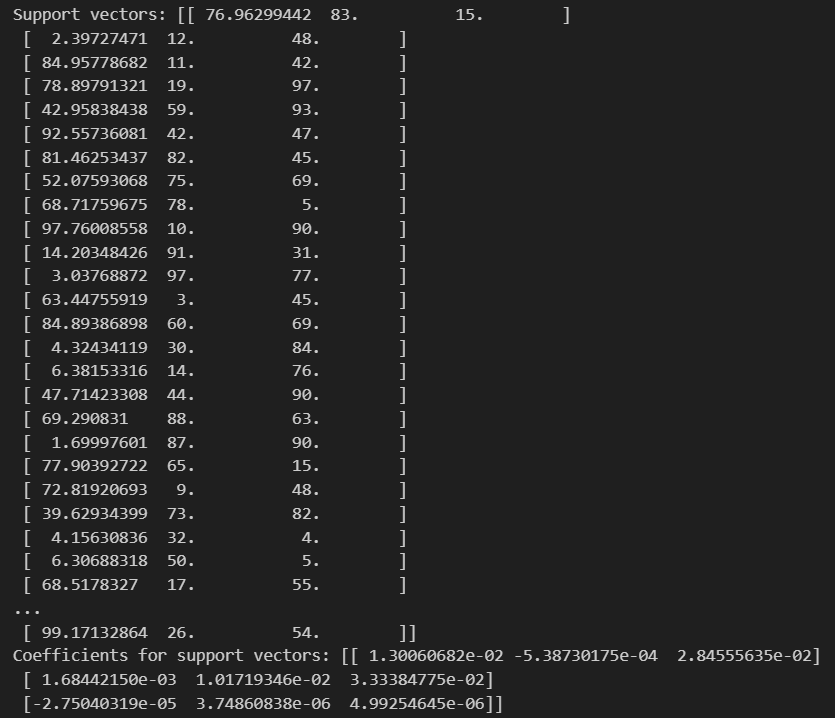
Линейная регрессия



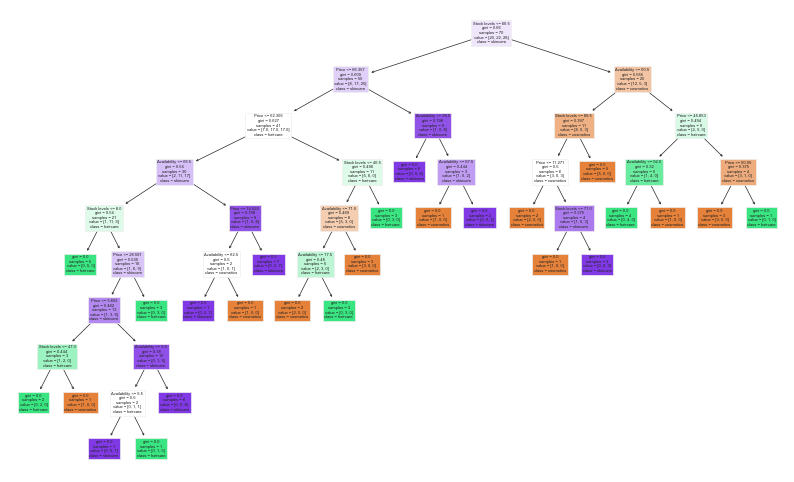
Полиномиальная регрессия



Вывод опорных векторов и коэффициентов для опорных векторов



Дерево решений



Качество моделей

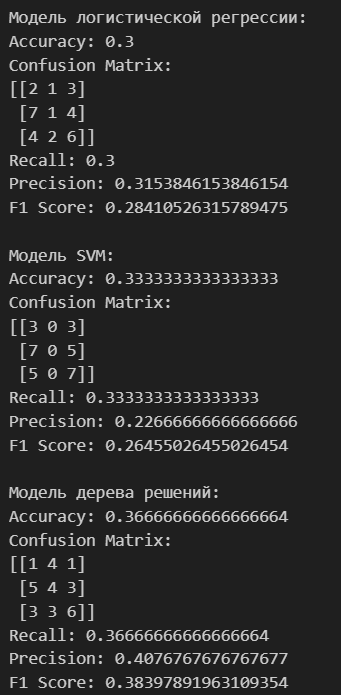


График важности признаков в дереве решений

