

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра технологий программирования

ОТЧЕТ по лабораторной 4,5
«АДЭФС. Лабораторная работа №4,5»

Зборовского Артёма Николаевича
студента 3 курса, 8 группы
Преподаватель
Полузеров Тимофей Дмитриевич

Минск 2025

1. ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ И КЛАСТЕРИЗАЦИЯ

1.1. Подготовка данных и выделение признаков

В качестве исходных данных использован файл `Annual 2005-2011.csv`, содержащий информацию о компаниях. Для анализа выделено 20 финансовых коэффициентов `k1`–`k20`. Для устранения влияния масштаба выполнена стандартизация признаков.

Код 1.1: Загрузка данных и стандартизация

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report

data = pd.read_csv(r"D:\python_task\data\Annual 2005-2011.csv")

coef_columns = [f"k{i}" for i in range(1, 21)]
features = data[coef_columns].copy()

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

1.2.

Кластеризация методом K-Mean

Для сегментации клиентов применен алгоритм K-Means. Количество кластеров установлено равным 4.

Код 1.2: Кластеризация K-Means

```
cluster_model = KMeans(n_clusters=4, random_state=12)
raw_clusters = cluster_model.fit_predict(scaled_features)

data["seg"] = raw_clusters
```

1.3. Ранжирование кластеров

Выполнено ранжирование кластеров.

Код 1.3: Ранжирование кластеров

```

cluster_profile_strength = (
    data.groupby("seg")[coef_columns].mean().mean(axis=1)
)

sorted_segments = cluster_profile_strength.sort_values(ascending=False).index

segment_mapping = {old: new_id for new_id, old in enumerate(sorted_segments, start=1)}

data["rating"] = data["seg"].map(segment_mapping)

```

1.4.

Формирование обучающей и тестовой выборки

Данные разделены на обучающую и тестовую выборки стратифицированным способом.

Код 1.4: Стратифицированное разделение данных

```

test_parts = []
for grp, block in data.groupby("rating"):
    part = block.sample(frac=0.10, random_state=12)
    test_parts.append(part)

test_split = pd.concat(test_parts, ignore_index=True)
train_split = data.drop(test_split.index)

X_train = train_split[coef_columns]
y_train = train_split["rating"]

X_test = test_split[coef_columns]
y_test = test_split["rating"]

```

2. ПОСТРОЕНИЕ И АНАЛИЗ МОДЕЛЕЙ КЛАССИФИКАЦИИ

2.1. Линейный дискриминантный анализ (LDA)

```

lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)

train_predictions = lda.predict(X_train)
test_predictions = lda.predict(X_test)

print("Дискриминантный анализ\n")

print("Точность (train):", accuracy_score(y_train, train_predictions))
print("Точность (test): ", accuracy_score(y_test, test_predictions))

print("\nМатрица ошибок (test):")
print(confusion_matrix(y_test, test_predictions))

```

2.2. Мультиномиальная логистическая регрессия

Код 2.2: Построение и оценка логистической регрессии

```
print("\nПодробный отчёт по классам:")
print(classification_report(y_test, test_predictions))

scaler2 = StandardScaler()
X_train_scaled = scaler2.fit_transform(X_train)
X_test_scaled = scaler2.transform(X_test)
logit_model = LogisticRegression(
    solver="lbfgs",
    max_iter=5000
)

logit_model.fit(X_train_scaled, y_train)
logit_predictions = logit_model.predict(X_test_scaled)

print("Мультиномиальная логит-модель\n")

print("Мультиномиальная логит-модель\n")

print("Точность (test):", accuracy_score(y_test, logit_predictions))

print("\nМатрица ошибок логит-модели:")
print(confusion_matrix(y_test, logit_predictions))

print("\nКлассификационный отчёт:")
print(classification_report(y_test, logit_predictions))

print("\nКоэффициенты модели (для каждого рейтинга):")
coeffs = pd.DataFrame(logit_model.coef_, columns=coef_columns)
coeffs["Класс"] = logit_model.classes_
print(coeffs)
```

3. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТА И АНАЛИЗ

3.1. Результаты выполнения программы:

дискриминантный анализ

Точность (train): 0.9076288659793814
Точность (test): 0.9037037037037037

Матрица ошибок (test):

```
[[ 20   3   3   0]
 [  0 102   1   0]
 [  0   8  71   1]
 [  0   5   5  51]]
```

Подробный отчёт по классам:

	precision	recall	f1-score	support
1	1.00	0.77	0.87	26
2	0.86	0.99	0.92	103
3	0.89	0.89	0.89	80
4	0.98	0.84	0.90	61
accuracy			0.90	270
macro avg	0.93	0.87	0.90	270
weighted avg	0.91	0.90	0.90	270

Мультиомиальная логит-модель

Точность (test): 0.9888888888888889

Матрица ошибок логит-модели:

```
[[ 26   0   0   0]
 [  0 102   1   0]
 [  1   1  78   0]
 [  0   0   0  61]]
```

Классификационный отчёт:

	precision	recall	f1-score	support
1	0.96	1.00	0.98	26
2	0.99	0.99	0.99	103
3	0.99	0.97	0.98	80
4	1.00	1.00	1.00	61
accuracy			0.99	270
macro avg	0.99	0.99	0.99	270
weighted avg	0.99	0.99	0.99	270

Коэффициенты модели (для каждого рейтинга):

	k1	k2	k3	k4	...	k18	k19	k20	Класс
0	3.195119	2.681909	3.212739	1.059539	...	1.182578	1.200872	1.052851	1
1	-0.293178	-1.108525	-0.756047	1.050908	...	0.478161	-0.527073	-0.540743	2
2	-0.708226	-0.976689	-0.771951	0.228494	...	0.957170	1.525086	1.372727	3
3	-2.193714	-0.596695	-1.684741	-2.338941	...	-2.617909	-2.198886	-1.884835	4