МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра теории вероятностей и математической статистики

ОТЧЕТ по лабораторной работе №2

Константинова Николая Андреевича
студента 3 курса, 8 группы

Преподаватель
Полузёров Тимофей Дмитриевич

Минск, 2025

Содержание

Часть 1. Применение факторного анализа к имеющимся данным

Проверяем целесообразность PCA с помощью корреляционной матрицы:

```
corr_matrix = np.corrcoef(X_scaled.T)
eigenvalues = np.linalg.eigvals(corr_matrix)

print(f"Определитель корреляционной матрицы: {np.linalg.det(corr_matrix):.6f}")
print(f"Количество собственных значений > 1: {sum(eigenvalues > 1)}")
print(f"Общая дисперсия: {np.sum(eigenvalues):.2f}")
```

Используем функцию PCA из библиотеки scikit-learn; для определения значимых компонент используем критерий Кайзера-Мейера-Олкина:

```
def calculate_kmo(X):
    corr_matrix = np.corrcoef(X.T)
    inv_corr_matrix = np.linalg.inv(corr_matrix)

    n_vars = X.shape[1]
    kmo_num = 0
    kmo_den = 0

    for i in range(n_vars):
        for j in range(n_vars):
            if i != j:
                kmo_num += corr_matrix[i,j]**2
                kmo_den += corr_matrix[i,j]**2 + (inv_corr_matrix[i,j]**2 if i != j else 0)

    return kmo_num / kmo_den if kmo_den != 0 else 0
```

и критерий сферичности Бартлетта:

```
def bartlett_sphericity_test(X):
    corr_matrix = np.corrcoef(X.T)
    n = X.shape[0]
    p = X.shape[1]
    det_corr = np.linalg.det(corr_matrix)
    chi_square = -((n - 1) - (2 * p + 5) / 6) * np.log(det_corr)
    df = p * (p - 1) / 2
    p_value = 1 - chi2.cdf(chi_square, df)
    return chi_square, df, p_value, det_corr
```

Результаты:

```
KMO-критерий: 0.000
Тест сферичности Бартлетта:
  Хи-квадрат: 39162.98
  Степени свободы: 253.0
  p-value: 0.000000
Определитель корреляционной матрицы: 0.000000
Общая дисперсия: 23.00
```

Теперь применяем PCA. Значимые компоненты определяем по критерию Кайзера.

```
kaiser_components = sum(pca.explained_variance_ > 1) #критерий Кайзера
variance_80 = np.where(cumulative_variance >= 0.8)[0]
components_80 = variance_80[0] + 1 if len(variance_80) > 0 else len(explained_variance)

print(f"\n Сводка по компонентам:")
print(f"По критерию Кайзера (λ > 1): {kaiser_components} компонент")
print(f"Для 80% дисперсии нужно: {components_80} компонент")
print(f"Первые 3 компоненты объясняют: {cumulative_variance[2]*100:.1f}% дисперсии")
```

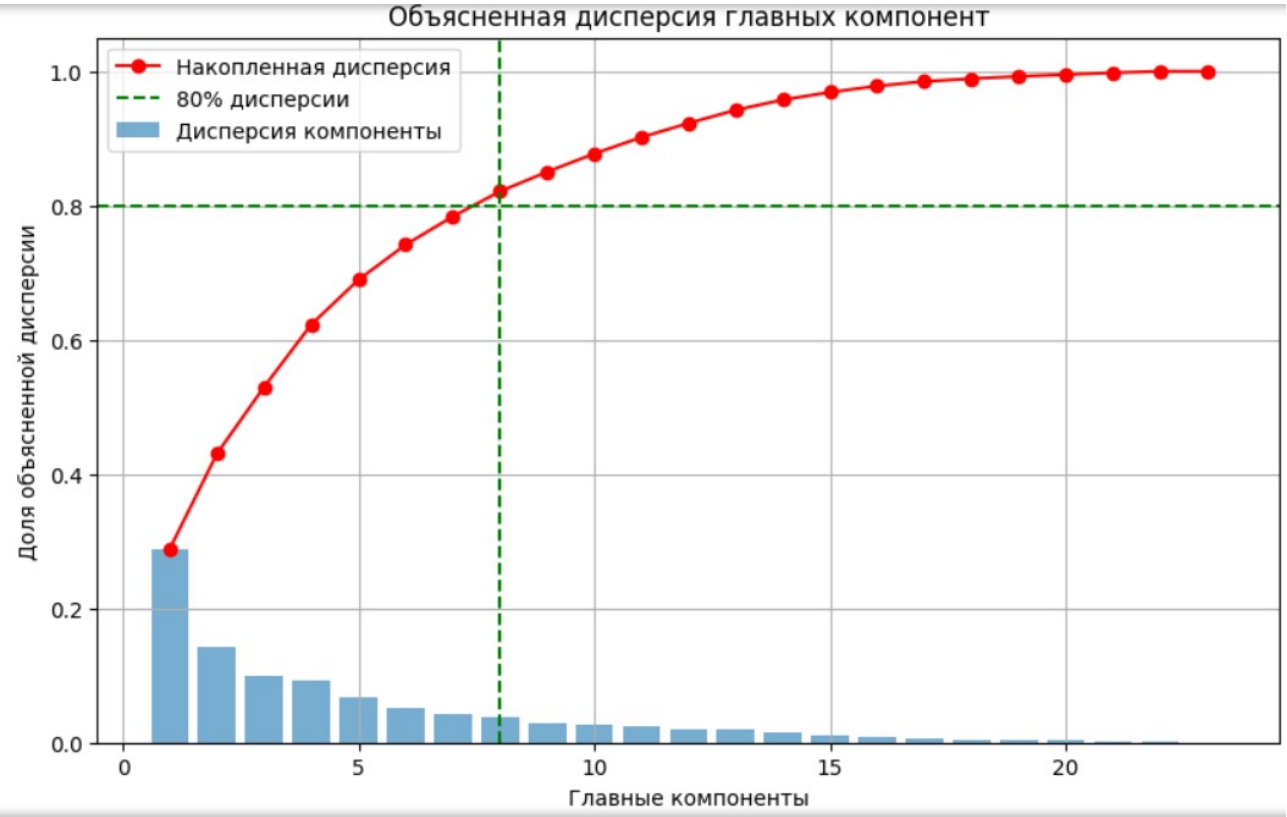
Результаты:

```
 РЕЗУЛЬТАТЫ PCA
Объясненная дисперсия по компонентам:
Компонента   Дисперсия    Накопленная
---------------------------------------------
PC1            0.2890 ( 28.90%) 0.2890 ( 28.90%)
PC2            0.1416 ( 14.16%) 0.4306 ( 43.06%)
PC3            0.0994 (  9.94%) 0.5300 ( 53.00%)
PC4            0.0929 (  9.29%) 0.6230 ( 62.30%)
PC5            0.0665 (  6.65%) 0.6895 ( 68.95%)
PC6            0.0523 (  5.23%) 0.7417 ( 74.17%)
PC7            0.0420 (  4.20%) 0.7837 ( 78.37%)
PC8            0.0374 (  3.74%) 0.8211 ( 82.11%)
PC9            0.0292 (  2.92%) 0.8502 ( 85.02%)
PC10           0.0271 (  2.71%) 0.8773 ( 87.73%)

 СВОДКА ПО КОМПОНЕНТАМ:
По критерию Кайзера (λ > 1): 6 компонент
Для 80% дисперсии нужно: 8 компонент
Первые 3 компоненты объясняют: 53.0% дисперсии
```

Часть 2. Определение наиболее значимых коэффициентов

Наиболее значимые коэффициенты — коэффициенты с наибольшим процентом дисперсии исходных показателей. Будем исследовать все дисперсии.



Анализ показывает, что, например, для 80% дисперсии необходимо 8 компонент.

Часть 3. Расчёт интегральных показателей кредитоспособности

Для расчета будем использовать компоненты, полученные в предыдущей части. Нормализуем веса и рассчитаем интегральный показатель, а потом нормализуем и его от 0 до 100:

```
n_components = components_80
selected_components = principal_components[:, :n_components]
selected_variance = explained_variance[:n_components]
#нормализуем веса (сумма=1)
weights = selected_variance / np.sum(selected_variance)
#расчет интегрального показателя
credit_score = np.zeros(principal_components.shape[0])
for i in range(n_components):
    credit_score += weights[i] * principal_components[:, i]
#нормировка от 0 до 100
credit_score_normalized = 100 * (credit_score - credit_score.min()) / (credit_score.max() - credit_score.min())
```

Результаты:

```
Использовано компонент: 8
Веса компонент:
  PC1: вес = 0.352 (28.9% дисперсии)
  PC2: вес = 0.172 (14.2% дисперсии)
  PC3: вес = 0.121 (9.9% дисперсии)
  PC4: вес = 0.113 (9.3% дисперсии)
  PC5: вес = 0.081 (6.7% дисперсии)
  PC6: вес = 0.064 (5.2% дисперсии)
  PC7: вес = 0.051 (4.2% дисперсии)
  PC8: вес = 0.046 (3.7% дисперсии)

Интегральный показатель кредитоспособности рассчитан:
Диапазон: 0.0 - 100.0
Среднее: 54.9
Стандартное отклонение: 17.4
```