



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)

Кафедра прикладной математики

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 5
по дисциплине «Технологии и инструментальный анализ
больших данных»

Выполнил студент группы ИКБО-20-21
Проверил ассистент кафедры ПМ ИИТ

Сидоров С.Д.
Тетерин Н.Н.

Москва 2024

Практическая работа

1. Найти данные для классификации. Данные в группе повторяться не должны. Предобработать данные, если это необходимо.

Листинг 1:

```
print("Задание 1: Загрузка и предобработка данных")
url = "https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.data"

# Определение названий колонок для датасета
columns = [
    'age', 'workclass', 'fnlwgt', 'education', 'education_num',
    'marital_status',
    'occupation', 'relationship', 'race', 'sex', 'capital_gain',
    'capital_loss',
    'hours_per_week', 'native_country', 'income'
];

# Загрузка данных в pandas DataFrame
df_real = pd.read_csv(url, header=None, names=columns, na_values=" ?",
sep=',\s', engine='python')

# Удаляем строки с пропущенными значениями
df_real = df_real.dropna()

# Преобразуем целевой признак (income) в бинарный формат
df_real['income'] = df_real['income'].apply(lambda x: 1 if x == '>50K'
else 0)

# Выбираем значимые признаки
X_real = df_real[['age', 'education_num', 'hours_per_week',
'capital_gain', 'capital_loss']]
y_real = df_real['income']

print("Данные успешно загружены и предобработаны\n")
```

2. Изобразить гистограмму, которая показывает баланс классов.

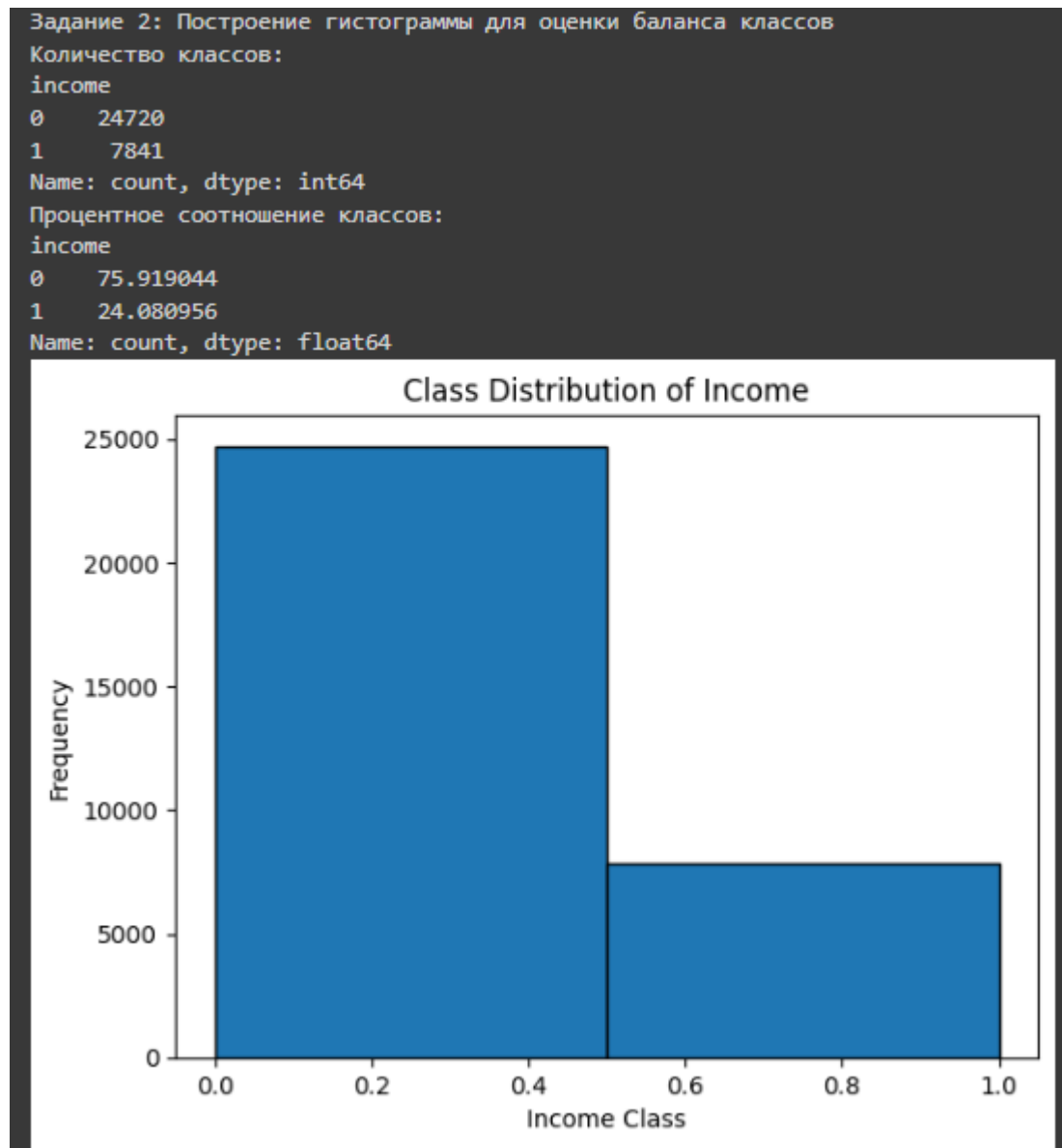
Сделать выводы.

Листинг 2:

```
class_counts = y_real.value_counts()
class_percentages = (class_counts / len(y_real)) * 100

print(f"Количество классов:\n{class_counts}")
print(f"Процентное соотношение классов:\n{class_percentages}")

plt.hist(y_real, bins=2, edgecolor='black')
plt.title("Class Distribution of Income")
plt.xlabel("Income Class")
plt.ylabel("Frequency")
plt.show()
```



Вывод:

Исходя из полученных данных и графиков можно сказать, что 24720 (76%) исследуемых зарабатывают меньше 50к\$ в год, 7841 (24%) - больше.

3. Разбить выборку на тренировочную и тестовую. Тренировочная для обучения модели, тестовая для проверки ее качества.

Листинг 4:

```
print("Задание 3: Разделение данных на тренировочную и тестовую выборки")
scaler_real = StandardScaler()
X_real_scaled = scaler_real.fit_transform(X_real)

X_train_real, X_test_real, y_train_real, y_test_real =
train_test_split(X_real_scaled, y_real, test_size=0.2,
random_state=42)
```

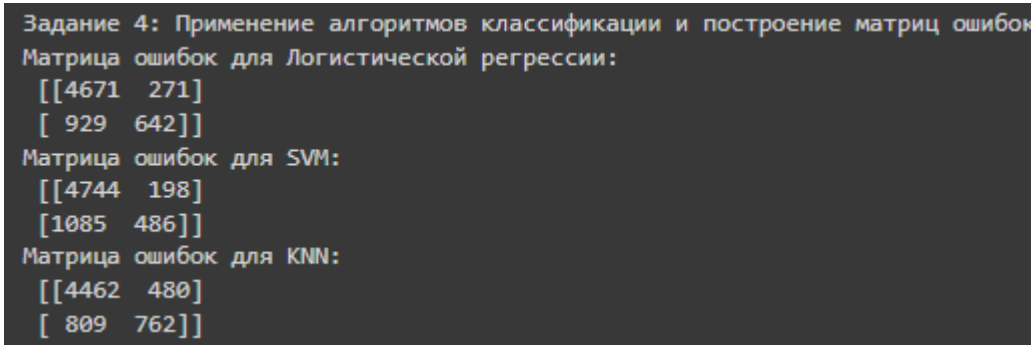
4. Применить алгоритмы классификации: логистическая регрессия, SVM, KNN. Построить матрицу ошибок по результатам работы моделей (использовать `confusion_matrix` из `sklearn.metrics`).

Листинг 5:

```
log_reg_real = LogisticRegression()
log_reg_real.fit(X_train_real, y_train_real)
y_pred_log_reg_real = log_reg_real.predict(X_test_real)
cm_log_reg_real = confusion_matrix(y_test_real, y_pred_log_reg_real)
print("Матрица ошибок для Логистической регрессии:\n",
      cm_log_reg_real)

# SVM
svm_real = SVC(kernel='linear')
svm_real.fit(X_train_real, y_train_real)
y_pred_svm_real = svm_real.predict(X_test_real)
cm_svm_real = confusion_matrix(y_test_real, y_pred_svm_real)
print("Матрица ошибок для SVM:\n", cm_svm_real)

# KNN
knn_real = KNeighborsClassifier(n_neighbors=5)
knn_real.fit(X_train_real, y_train_real)
y_pred_knn_real = knn_real.predict(X_test_real)
cm_knn_real = confusion_matrix(y_test_real, y_pred_knn_real)
print("Матрица ошибок для KNN:\n", cm_knn_real)
```



```
Задание 4: Применение алгоритмов классификации и построение матриц ошибок
Матрица ошибок для Логистической регрессии:
[[4671  271]
 [ 929  642]]
Матрица ошибок для SVM:
[[4744  198]
 [1085  486]]
Матрица ошибок для KNN:
[[4462  480]
 [ 809  762]]
```

5. Сравнить результаты классификации, используя ассигурацию, precision, recall и f1-меру (можно использовать `classification_report` из `sklearn.metrics`). Сделать выводы.

Листинг 6:

```
print("Отчет по Логистической регрессии:\n")
print(classification_report(y_test_real, y_pred_log_reg_real))

# SVM
print("Отчет по SVM:\n")
print(classification_report(y_test_real, y_pred_svm_real))

# KNN
print("Отчет по KNN:\n")
print(classification_report(y_test_real, y_pred_knn_real))
```

Задание 5: Сравнение моделей по метрикам

Отчет по Логистической регрессии:

	precision	recall	f1-score	support
0	0.83	0.95	0.89	4942
1	0.70	0.41	0.52	1571
accuracy			0.82	6513
macro avg	0.77	0.68	0.70	6513
weighted avg	0.80	0.82	0.80	6513

Отчет по SVM:

	precision	recall	f1-score	support
0	0.81	0.96	0.88	4942
1	0.71	0.31	0.43	1571
accuracy			0.80	6513
macro avg	0.76	0.63	0.66	6513
weighted avg	0.79	0.80	0.77	6513

Отчет по KNN:

	precision	recall	f1-score	support
0	0.85	0.90	0.87	4942
1	0.61	0.49	0.54	1571
accuracy			0.80	6513
macro avg	0.73	0.69	0.71	6513
weighted avg	0.79	0.80	0.79	6513

1. Логистическая регрессия

Матрица ошибок:

Верно предсказаны классы 0: 4671

Ошибочно предсказаны классы 0 как 1: 271

Верно предсказаны классы 1: 642

Ошибочно предсказаны классы 1 как 0: 929

Precision (точность):

Для класса 0 — 83%

Для класса 1 — 70%

Это означает, что модель относительно точно предсказывает оба класса, но делает больше ошибок в классе 1.

Recall (полнота):

Для класса 0 — 95%

Для класса 1 — 41%

Модель лучше распознает класс 0 (более 95%), но имеет проблемы с распознаванием класса 1 (только 41%).

F1-score:

Для класса 0 — 0.89

Для класса 1 — 0.52

Модель более сбалансирована по классу 0, но по классу 1 недостаточно хорошо.

Вывод: Логистическая регрессия показывает хорошее качество предсказаний для класса 0, но модель плохо справляется с распознаванием класса 1. Для задач, где критично корректно предсказать класс 1, эта модель может быть недостаточно эффективной.

2. SVM (Метод опорных векторов)

Матрица ошибок:

Верно предсказаны классы 0: 4744

Ошибочно предсказаны классы 0 как 1: 198

Верно предсказаны классы 1: 486

Ошибочно предсказаны классы 1 как 0: 1085

Precision:

Для класса 0 — 81%

Для класса 1 — 71%

Модель показывает более высокую точность для класса 1 по сравнению с Логистической регрессией.

Recall:

Для класса 0 — 96%

Для класса 1 — 31%

Хотя SVM лучше справляется с правильной идентификацией класса 0, распознавание класса 1 страдает (всего 31%).

F1-score:

Для класса 0 — 0.88

Для класса 1 — 0.43

F1 для класса 1 хуже, чем у Логистической регрессии.

Вывод: SVM показывает лучшие результаты по точности для класса 1, но значительно страдает по полноте для этого же класса. В результате модель чаще упускает класс 1, что может быть критично для задач с несбалансированными данными.

3. KNN (Метод ближайших соседей)

Матрица ошибок:

Верно предсказаны классы 0: 4462

Ошибочно предсказаны классы 0 как 1: 480

Верно предсказаны классы 1: 762

Ошибочно предсказаны классы 1 как 0: 809

Precision:

Для класса 0 — 85%

Для класса 1 — 61%

Модель делает больше ошибок при предсказании класса 1, хотя точность для класса 0 высокая.

Recall:

Для класса 0 — 90%

Для класса 1 — 49%

Модель лучше распознает класс 1 по сравнению с Логистической регрессией и SVM, но все равно полнота не идеальна.

F1-score:

Для класса 0 — 0.87

Для класса 1 — 0.54

F1-score для класса 1 лучше, чем у других моделей.

Вывод: KNN показывает сбалансированные результаты для обоих классов по сравнению с Логистической регрессией и SVM. Он имеет лучшую полноту по классу 1, но делает больше ошибок по сравнению с другими моделями.

Итоговое сравнение

Логистическая регрессия имеет наилучшую полноту по классу 0, но страдает при предсказании класса 1.

SVM обеспечивает лучшую точность для класса 1, но его полнота для этого класса слишком низка.

KNN является более сбалансированной моделью, так как он лучше справляется с предсказанием обоих классов, хотя точность и полнота по классу 1 всё ещё ниже оптимальной.

KNN может быть предпочтительным выбором, если важна сбалансированность предсказаний для обоих классов, в то время как Логистическая регрессия и SVM лучше подходят для задач, где критична высокая точность для класса 0.