|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 инструментарий анализа больших данных»** | |
|  | |
| Выполнил студент группы ИКБО-20-21 | Сидоров С.Д. |
| Проверил ассистент кафедры ПМ ИИТ | Тетерин Н.Н. |

Москва 2024

**Практическая работа**

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

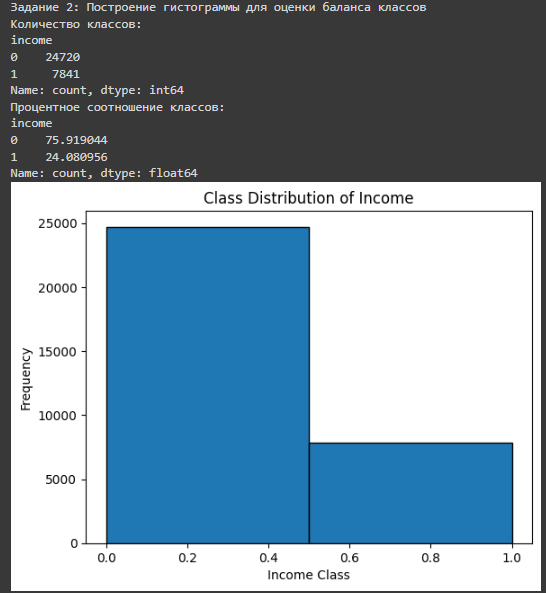
Листинг 1:

|  |
| --- |
| 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") |

1. 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 зараюатывают меньше 50к$ в год, 7841 (24%) - больше.

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

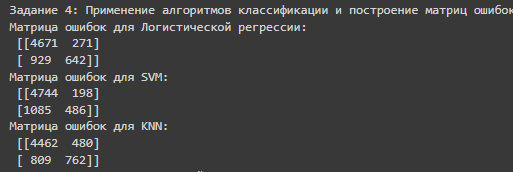
Листинг 4:

|  |
| --- |
| 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) |

1. Применить алгоритмы классификации: логистическая регрессия, SVM, KNN. Построить матрицу ошибок по результатам работы моделей (использовать confusion\_matrix из sklearn.metrics).

Листинг 5:

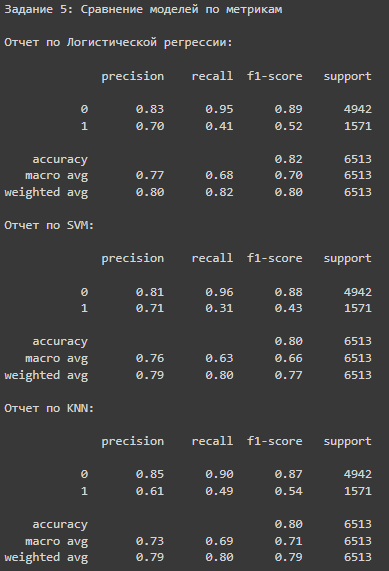
|  |
| --- |
| log\_reg\_real = LogisticRegression()  log\_reg\_real.fit(X\_train\_real, y\_train\_real)  y\_pred\_log\_reg\_real = log\_reg\_real.predict(X\_test\_real)  cm\_log\_reg\_real = confusion\_matrix(y\_test\_real, y\_pred\_log\_reg\_real)  print("Матрица ошибок для Логистической регрессии:\n", 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) |



1. Сравнить результаты классификации, используя accuracy, 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)) |



* + - 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.