Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 Прикладная информатика**

ОТЧЕТ

по лабораторной работе № 1
Вариант 1

Название:

Дисциплина: Прикладной анализ данных

Студент ИУ6-55Б _____ А.Д. Шевченко

(Подпись, дата) (И.О. Фамилия)

Преподаватель _____ М.А. Кулаев

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

1. Разделение данных на обучающую (85%) и тестовую часть (15%) случайным образом (можно сделать более корректным методом – разделить в такой пропорции с сохранением распределения таргета в каждой подвыборке – **желательно, но не обязательно**).

2. Нормирование (масштабирование) исходных данных. Обратите внимание,что данные для нормализации (масштабирования) рассчитываются только на основе обучающей выборки.

3. С помощью библиотеки sklearn сделать fit-predict модели kNN. Перебрать по сетке параметр числа соседей с целью определения наилучшего на тестовой выборке.

4. С помощью библиотеки sklearn сделать fit-predict модели логистической регрессии. Перебрать по сетке параметр регуляризации с целью определения наилучшего на тестовой выборке.

5. С помощью библиотеки sklearn сделать fit-predict модели дерева решений. Перебрать по сетке параметр глубины дерева с целью определения наилучшего на тестовой выборке. **Дополнительно (желательно, но не обязательно):** с помощью библиотеки sklearn сделать fit-predict модели случайного леса. Перебрать по сетке параметр глубины дерева с целью определения наилучшего на тестовой выборке.

6. Сравнить качество всех моделей на обучающей и тестовой выборке отдельно по метрикам Accuracy, ROC-AUC, Precision, Recall, F1-мера. Обратите внимание, что 4 из 5 метрик требуют определения порога отсечения по вероятности. В качестве эвристики предлагается взять его как среднее значение полученных вероятностей (**желательно, но не обязательно:**подобрать по сетке такой порог, при котором precision и recall примерно уравниваются).

7. Проанализировать различие в качестве между моделями. Определить на основе метрик модели, в которых сильно выражено переобучение.

8. Сравнить полученную важность признаков в модели логистической регрессии, в модели деревьев решений и в случайном лесе (для древесных моделей это можно сделать с помощью ключа feature_importances у обученной модели). Проинтерпретировать полученную важность признаков.

Задание 1. Разделение данных на выборки.

В соответствии с номером варианта были подготовлены исходные данные – строки, соответствующие регионам: Северный, Северо-Западный, Центральный, Волго-Вятский, Центрально-Черноземный, Северо-Кавказский, Уральский районы. Далее из этих строк был составлен новый Excel-файл, который был прочитан и записан в соответствующие переменные X и Y (рисунок 1).

```
[7]  import pandas as pd
     import numpy as np
     from google.colab import files

[5]  uploaded = files.upload()
     df = pd.read_excel('ADA_LR2_source.xlsx')

[21] X = df.iloc[:, :9]
     Y = df.iloc[:, 9]

     X.head()
```

	x1	x2	x3	x4	x5	x6	x7	x8	x9
0	8.5	16.3	6.8	5.6	17.4	163	151	23.6	2344
1	9.3	12.6	7.2	5.5	25.3	194	239	9.2	1809
2	8.7	4.6	6.5	4.2	16.2	152	192	26.9	2406
3	8.6	6.2	6.1	4.0	17.4	190	205	20.1	2023
4	8.1	11.4	7.7	6.4	5.9	183	198	22.0	1419

Рисунок 1 – загрузка данных.

Для разделения исходных данных на тренировочные и тестовые можно разбить их случайным образом, но есть и другой вариант – воспользоваться готовым методом train_test_split библиотеки sklearn. Воспользуемся второй опцией. Также для равномерного распределения

классов применим стратификацию в соответствии с распределением по классам Y.

```
random_seed = 27
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.85, random_state=random_seed, stratify=Y)
print('Распределение Y_train по классам 0 и 1 соответственно:', np.bincount(Y_train))
print('Распределение Y_test по классам 0 и 1 соответственно:', np.bincount(Y_test))

Распределение Y_train по классам 0 и 1 соответственно: [18 14]
Распределение Y_test по классам 0 и 1 соответственно: [3 3]
```

Рисунок 2 – разделение исходных данных на две выборки.

Задание 2. Нормирование данных

Далее было осуществлено нормирование данных. Для этого была выбрана Z-нормализация (рисунок 3). Также стоит обратить внимание на то, что для тестовых данных в формулу необходимо подставить mean_train и deviation_train.

```
# Z-нормализация данных

mean_train = np.mean(X_train, axis=0)
deviation_train = np.std(X_train, axis=0)

X_train_norm = (X_train - mean_train) / deviation_train
X_test_norm = (X_test - mean_train) / deviation_train
```

Рисунок 3 - нормирование данных.

Задание 3. С помощью библиотеки sklearn сделать fit-predict модели kNN. Перебрать по сетке параметр числа соседей с целью определения наилучшего на тестовой выборке.

Для выполнения задания воспользуемся методом GridSearchCV библиотеки sklearn, который находит наилучшие параметры путем

простого перебора – он создает по модели на каждую возможную комбинацию параметров. Fit-predict модели KNN изображен на рисунке 4.

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report

knn = KNeighborsClassifier()
grid_params = {'n_neighbors': range(1, 26)}
grid = GridSearchCV(knn, grid_params, cv=5, scoring='accuracy')
knn_model = grid.fit(X_train_norm, Y_train)

print('Наилучший параметр (число соседей):', knn_model.best_params_)
print('Точность (без тестовых данных):', knn_model.best_score_ * 100)

knn_model_best = knn_model.best_estimator_
Y_predicted = knn_model_best.predict(X_test_norm)

print(classification_report(Y_test, Y_predicted))
```

```
Наилучший параметр (число соседей): {'n_neighbors': 1}
Точность (без тестовых данных): 63.33333333333333
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         3
           1       1.00      1.00      1.00         3

    accuracy                           1.00         6
   macro avg       1.00      1.00      1.00         6
weighted avg       1.00      1.00      1.00         6
```

Рисунок 4 – fit-predict модели KNN.

Поясним параметры, передающиеся в GridSearchCV:

- KNeighborsClassifier – метод обучения;
- neighbors (range(1, 26)) – число соседей для перебора;
- scoring (accuracy) – означает, что наилучший параметр будет выбираться, основываясь на точности;
- cv (5) – часто использующееся значение для кросс-валидации (позволяет выявить проблему переобучения и принять меры по её устранению).

Однако точность предсказаний достаточно низка. Попробуем изменить random_seed и установим новое значение, равное 12. В таком случае результаты следующие (рисунок 5):

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification_report

knn = KNeighborsClassifier()
grid_params = {'n_neighbors': range(1, 26)}
grid = GridSearchCV(knn, grid_params, cv=5, scoring='accuracy')
knn_model = grid.fit(X_train_norm, Y_train)

print('Наилучший параметр (число соседей):', knn_model.best_params_)
print('Точность (без тестовых данных):', knn_model.best_score_ * 100)

knn_model_best = knn_model.best_estimator_
Y_predicted = knn_model_best.predict(X_test_norm)

print(classification_report(Y_test, Y_predicted))
```

```
Наилучший параметр (число соседей): {'n_neighbors': 1}
Точность (без тестовых данных): 63.33333333333334
              precision    recall  f1-score   support

           0       1.00      0.33      0.50         3
           1       0.60      1.00      0.75         3

    accuracy                           0.67         6
   macro avg       0.80      0.67      0.62         6
weighted avg       0.80      0.67      0.62         6
```

Рисунок 5 – fit-predict модели KNN после изменения значения random_seed.

По выведенным результатам можно судить о том, что модель показала наивысшую точность на тренировочных данных (63.33%) при числе соседей, равном 1. Рассмотрим классификационный отчет:

- precision – точность прогнозов модели; равна количеству правильно предсказанных значений, деленному на количество

всех предсказаний; в нашем случае 100% предсказаний для класса 0 и 60% предсказаний для класса 1 оказались верными;

- recall – доля положительных предсказаний модели, оказавшихся верными, равняется количеству правильно предсказанных положительных значений, деленному на сумму количества правильно предсказанных положительных значений и количества неверно предсказанных отрицательных значений;
- f1-score – гармоническое среднее между точностью (precision) и полнотой (recall); значение 0 означает совершенно бесполезную модель, 1 – идеальную модель; в нашем случае значение этого параметра равно 0.5 для класса 0 и 0.75 для класса 1;
- support – показывает количество образцов в каждом классе в тестовом наборе данных; в нашем случае 3 образца принадлежали к классу 0, 3 образца – к классу 1;
- accuracy – общая точность модели; в нашем случае она равна 0.67, что означает, что 67% значений были предсказаны верно.

Задание 4. С помощью библиотеки sklearn сделать fit-predict модели логистической регрессии. Перебрать по сетке параметр регуляризации с целью определения наилучшего на тестовой выборке.

В документации библиотеки sklearn указано, что лучшим методом решения логистической регрессии в случае работы с небольшим количеством данных является liblinear, что как раз подходит в нашем случае. Также в документации сообщается, что для этого метода доступны следующие методы регуляризации: L1 и L2.

Fit-predict модели логистической регрессии изображен на рисунке 5.

```
from sklearn.linear_model import LogisticRegression

logistic_regression = LogisticRegression()
grid_params = {'penalty': ['l1', 'l2'], 'solver': ['liblinear']}

grid = GridSearchCV(logistic_regression, grid_params, cv=5, scoring='accuracy')
logistic_model = grid.fit(X_train_norm, Y_train)

print('Наилучший параметр (метод решения):', grid.best_params_)
print('Точность (без тестовых данных):', grid.best_score_ * 100)

logistic_model_best = logistic_model.best_estimator_
Y_predicted = logistic_model_best.predict(X_test_norm)

print(classification_report(Y_test, Y_predicted))
```

```
Наилучший параметр (метод решения): {'penalty': 'l2', 'solver': 'liblinear'}
Точность (без тестовых данных): 68.0952380952381
              precision    recall  f1-score   support

           0       0.67      0.67      0.67         3
           1       0.67      0.67      0.67         3

    accuracy                           0.67         6
   macro avg       0.67      0.67      0.67         6
weighted avg       0.67      0.67      0.67         6
```

Рисунок 5 – fit-predict модели логистической регрессии.

Как показывает отчет, лучшим методом решения в нашей ситуации оказался метод L2.

Задание 5. С помощью библиотеки sklearn сделать fit-predict модели дерева решений. Перебрать по сетке параметр глубины дерева с целью определения наилучшего на тестовой выборке. Дополнительно (желательно, но не обязательно): с помощью библиотеки sklearn сделать fit-predict модели случайного леса. Перебрать по сетке параметр глубины дерева с целью определения наилучшего на тестовой выборке.

В качестве переменного параметра будет выступать max_depth – максимальная глубина дерева.

```
from sklearn.tree import DecisionTreeClassifier

decision_tree = DecisionTreeClassifier(splitter='best')
grid_params = {'max_depth': range(1, 30)}

grid = GridSearchCV(decision_tree, grid_params, cv=5, scoring='accuracy')
decision_tree_model = grid.fit(X_train_norm, Y_train)

print('Наилучший параметр (глубина дерева):', grid.best_params_)
print('Точность (без тестовых данных):', grid.best_score_ * 100)

decision_tree_model_best = decision_tree_model.best_estimator_
Y_predicted = decision_tree_model_best.predict(X_test_norm)

print(classification_report(Y_test, Y_predicted))
```

```
Наилучший параметр (глубина дерева): {'max_depth': 1}
Точность (без тестовых данных): 55.714285714285715
              precision    recall  f1-score   support

           0       0.75      1.00      0.86         3
           1       1.00      0.67      0.80         3

    accuracy                           0.83         6
   macro avg       0.88      0.83      0.83         6
weighted avg       0.88      0.83      0.83         6
```

Алгоритм показал, что лучшая максимальная глубина дерева будет равна 1.

Проделаем то же самое, но для модели случайного леса.

```
from sklearn.ensemble import RandomForestClassifier

random_forest = RandomForestClassifier()
grid_params = {'max_depth': range(1, 30)}

grid = GridSearchCV(random_forest, grid_params, cv=5, scoring='accuracy')
random_forest_model = grid.fit(X_train_norm, Y_train)

print('Наилучший параметр (глубина леса):', grid.best_params_)
print('Точность без тестовых данных:', random_forest_model.best_score_ * 100)

random_forest_model_best = random_forest_model.best_estimator_
Y_predicted = random_forest_model_best.predict(X_test_norm)

print(classification_report(Y_test, Y_predicted))
```

```
Наилучший параметр (глубина леса): {'max_depth': 11}
Точность без тестовых данных: 59.999999999999986
              precision    recall  f1-score   support

           0       0.75      1.00      0.86         3
           1       1.00      0.67      0.80         3

    accuracy                           0.83         6
   macro avg       0.88      0.83      0.83         6
weighted avg       0.88      0.83      0.83         6
```

Рисунок 8 - случайный лес.

Как мы можем видеть из отчета, лучшей максимальной глубиной дерева будет 11.

Задание 6-7. Сравнение качества моделей.

Построим график зависимости F1-меры от выбранного порога отсечения для каждой модели. Порог отсечения будет переменным от 0 до 1 с шагом 0.01. На рисунке 9 изображен код (для каждой модели будет меняться единственная строчка – та, что содержит имя модели). На рисунках 10-13 изображены графики для каждой модели.

```
from matplotlib import pyplot as plt
from sklearn.metrics import f1_score

thresholds = np.linspace(0, 1, 1000)
f1_values = []
Y_prob_values = knn_model_best.predict_proba(X_test)[:, 1]
Y_predicted = []

for threshold in thresholds:
    for value in Y_prob_values:
      if value >= threshold:
        Y_predicted.append(1)
      else:
        Y_predicted.append(0)
    f1 = f1_score(Y_test, Y_predicted)
    f1_values.append(f1)
    Y_predicted = []


plt.figure(figsize=(10, 6))
plt.plot(thresholds, f1_values)
plt.xlabel('Порог отсечения')
plt.ylabel('F1-мера')
plt.grid(True)
plt.title('Зависимость F1-меры от значения порога (DT)')
plt.xticks(np.arange(0, 1.05, 0.05))
plt.show()
```

Рисунок 9 – код построения графика.

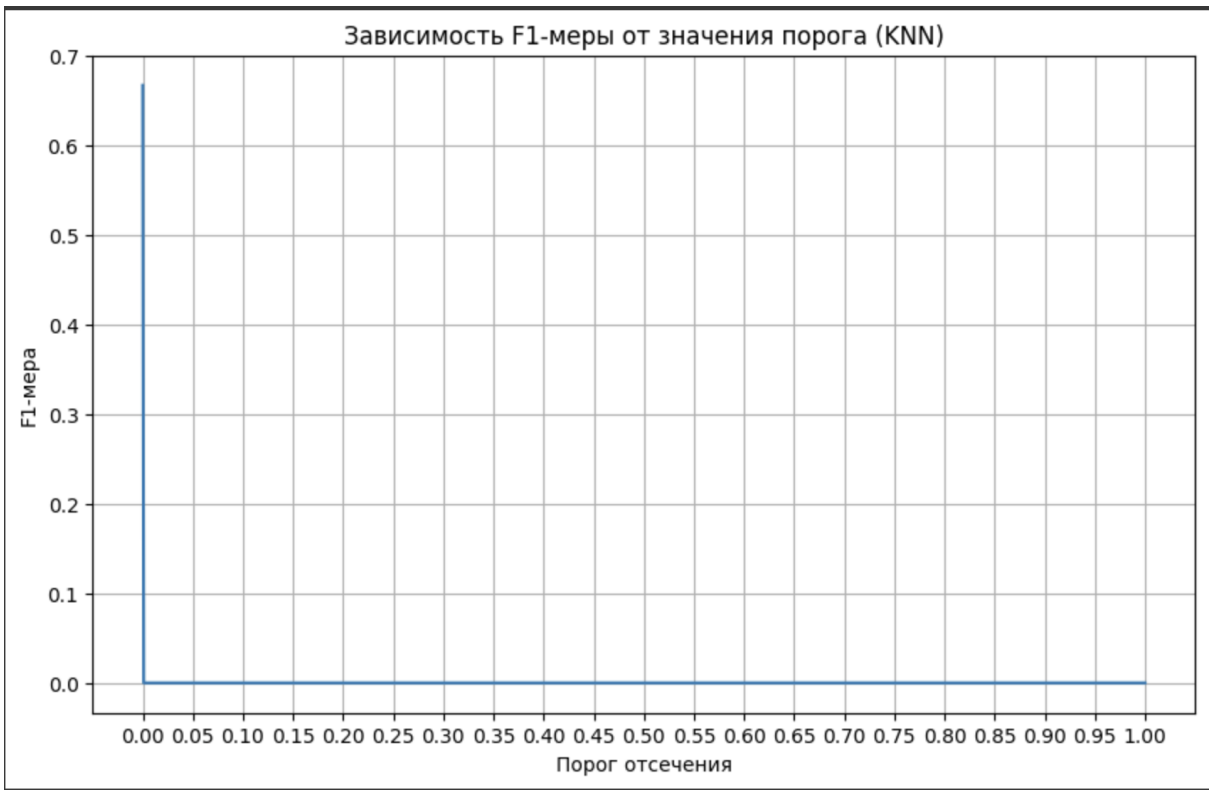

Рисунок 10 - график для модели KNN.

Судя по графику, лучшим порогом отсечения для модели KNN будет 0.

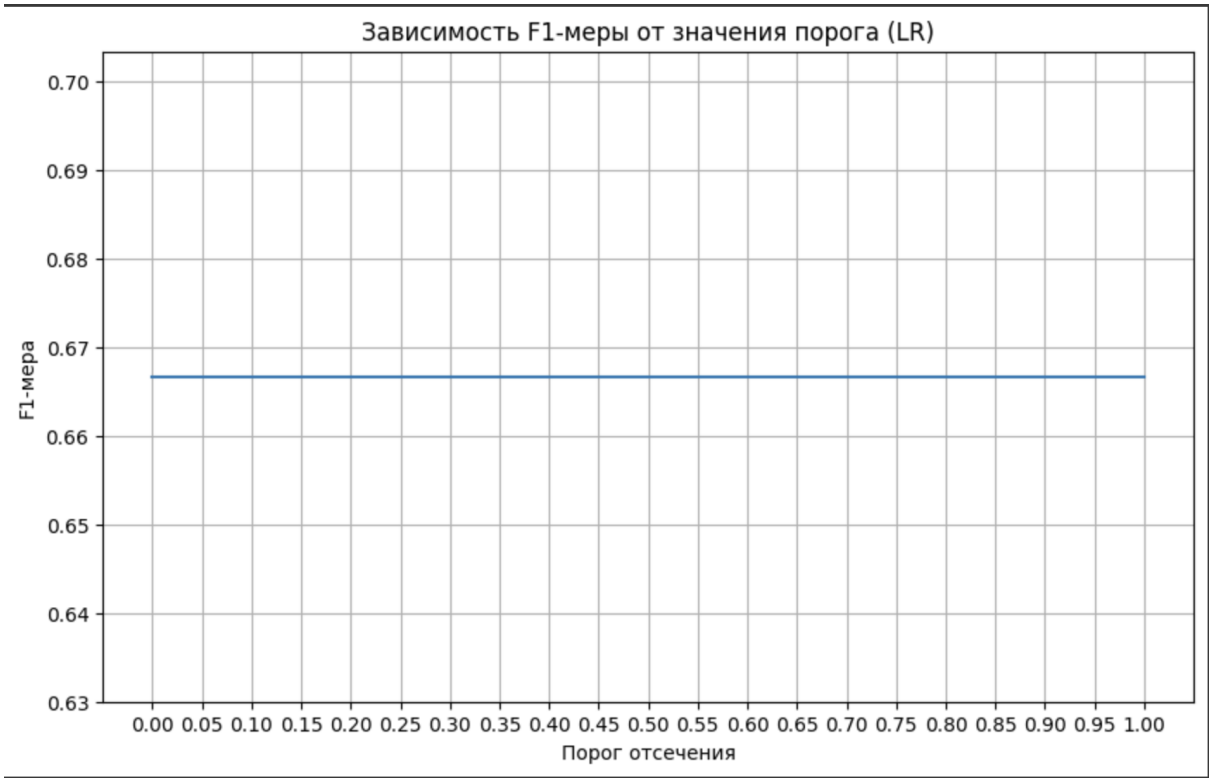

Рисунок 11 - график для модели логистической регрессии.

Здесь можно взять любой порог отсечения, так как при всех его значениях F1-мера постоянна.

Рисунок 12 - график для модели дерева решения.

Пусть порог отсечения будет равен 0.3 для дерева решений.

Рисунок 13 - график для модели случайного леса.

Пусть порог отсечения будет равен 0.45 для модели случайного леса.

Выведем параметры Accuracy, Precision, Recall, F1 и ROC-AUC для каждой модели на тестовых и тренировочных данных с учетом полученных порогов отсечения.

Таблица для тестовых данных:

	KNN	ЛГ	ДР	СЛ
Accuracy	0.67	0.67	0.67	0.83
Precision	0.6	0.67	0.6	1
Recall	1	0.67	1	0.67
F1	0.75	0.67	0.75	0.8
ROC-AUC	0.67	0.67	0.67	0.83

Вывод:

СЛ показал лучшие результаты, остальные модели находятся примерно на одном уровне.

Теперь посчитаем все те же метрики на основе обучающих данных.

	KNN	ЛР	ДР	СЛ
Accuracy	0.91	0.84	0.72	1
Precision	0.87	0.85	0.86	1
Recall	0.93	0.79	0.43	1
F1	0.9	0.81	0.57	1
ROC-AUC	0.91	0.84	0.69	1

Переобучение — явление, когда построенная модель хорошо объясняет примеры из обучающей выборки, но относительно плохо

работает на примерах, не участвовавших в обучении (на примерах из тестовой выборки).

Рассмотрим отдельно каждую модель и определим изменение метрик на тестовых данных, в отличии от обучающих. + вырасло, - упало, = равно

	KNN	ЛР	ДР	СЛ
Accuracy	-	-	-	-
Precision	-	-	-	=
Recall	+	-	+	-
F1	-	-	+	-
ROC-AUC	-	-	-	-

Исходя из полученных результатов можно сделать вывод, что проблема переобучения явно выражена в моделях логистической регрессии и KNN, так как показатели всех метрик на тестовых данных сильно ниже, чем на обучающих.

В остальных моделях лишь некоторые показатели оказались незначительно хуже, либо такими же.

Задание 8. Сравнить полученную важность признаков в модели логистическоӥ регрессии, в модели деревьев решениӥ и в случайном лесе (для древесных моделеӥ это можно сделать с помощью ключа feature_importances у обученноӥ модели). Проинтерпретировать полученную важность признаков.

В модели логистической регрессии важность признаков обычно определяется на основе абсолютных значений коэффициентов(весов), присвоенных каждому признаку. Чем больше абсолютное значение коэффициента, тем более важным считается соответствующий признак для модели. Если признак имеет положительный коэффициент, это означает, что этот признак положительно влияет на прогнозы модели. Увеличение значений этого признака будет увеличивать вероятность положительного

результата. Если же отрицательный коэффициент, то увеличение значений такого признака будет уменьшать вероятность положительного результата.

```
logistic_model_best.coef_[0]
```

```
array([ 0.31399521, -0.50555733,  1.1707514 , -0.70196637, -0.13216789,
        0.02303455, -0.35202038, -0.37467281,  0.57165752])
```

Рисунок 14 - веса логистической регрессии.

Наиболее значимыми признаками стали:

- X2 – смертность населения на 1000 человек
    - Высокая смертность может свидетельствовать о проблемах в системе здравоохранения и, может быть, недостаточно высоком уровне жизни населения
- X3 – число браков на 1000 человек
    - Большое число браков на 1000 человек может свидетельствовать о том, что будущие семейные пары скорее всего имеют постоянный источник дохода и могут позволить себе содержание новообразовавшейся семьи
- X4 – соотношении денежного дохода и прожиточного минимума
    - По идее, этот коэффициент должен быть положительным, так как он благоприятно влияет на качество и продолжительность жизни людей, поскольку сигнализирует о том, что люди могут позволить себе приобрести больше «приятных» необязательных товаров
- X9 – число зарегистрированных преступлений на 100000 населения
    - По идее, этот коэффициент должен быть отрицательным, так как большое количество зарегистрированных преступлений на 100000 населения указывает на проблемы в сфере правопорядка

Общий вывод: Исходя из полученных наиболее значимых признаков, можно сделать вывод о том, что на данном наборе данных социальные и экономические аспекты, такие как стабильность семейных отношений, здравоохранение и уровень преступности, оказывают большое влияние на среднюю продолжительность жизни мужчин в регионе.

Важность признаков в модели дерева решений можно оценить с помощью метода feature_importances_ после обучения модели. Этот метод позволяет узнать важность каждого признака для модели.

Интерпретация важности признаков в деревьях решений, как правило, основывается на двух факторах:

1. **Частота использования признака для разделения данных**: когда дерево решений строится, оно принимает решения о разделении данных на основе различных признаков. Признаки, которые чаще используются для разделения данных, могут считаться более важными.
2. **Улучшение качества прогнозов**: для каждого разделения, дерево решений оценивает, насколько успешно это разделение улучшает качество прогнозов.

```
feature_importance = decision_tree_model.best_estimator_.feature_importances_
plt.figure(figsize=(10, 6))
plt.bar(range(len(feature_importance)), feature_importance)
plt.xticks(np.arange(0, 9, 1))
plt.xlabel('Признаки')
plt.ylabel('Важность признаков')
plt.title('Важность признаков в модели дерева решений')
plt.show()
```

Рисунок 15 – Построение графика важности признаков в дереве решений

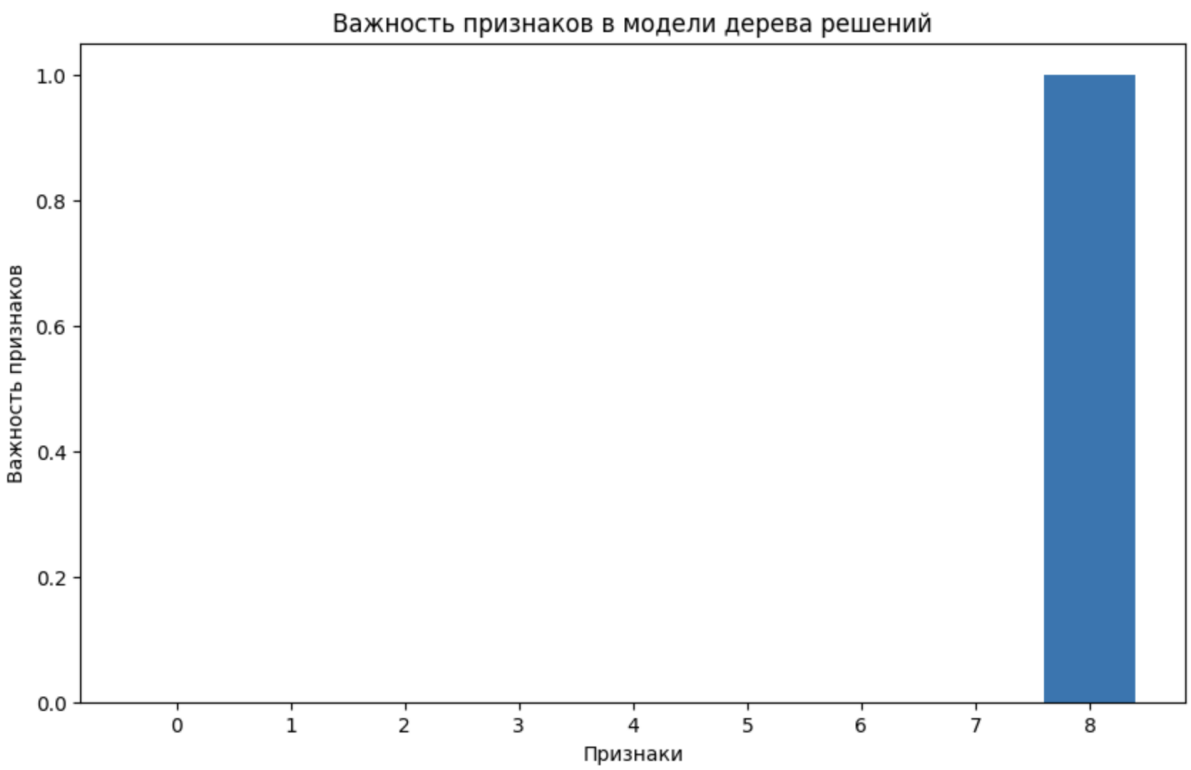

Рисунок 16 - Важность признаков в дереве решений

В результате получился один важный признак:

- X9 – число зарегистрированных преступлений на 100000 населения

Чтобы выявить самые важные признаки для модели случайного леса, используем тот же самый метод (feature_importances_).

Однако в случайных лесах это делается путем усреднения важности признаков по всем деревьям в лесу. Это усреднение делает оценку важности более стабильной и надежной, чем при использовании одиночного дерева решений.

```
feature_importance = forest_model.best_estimator_.feature_importances_
plt.figure(figsize=(10, 6))
plt.bar(range(len(feature_importance)), feature_importance)
plt.xticks(np.arange(0, 9, 1))
plt.xlabel('Признаки')
plt.ylabel('Важность признаков')
plt.title('Важность признаков в модели случайного леса')
plt.show()
```

Рисунок 17 – Построение графика важности признаков для модели случайного леса.

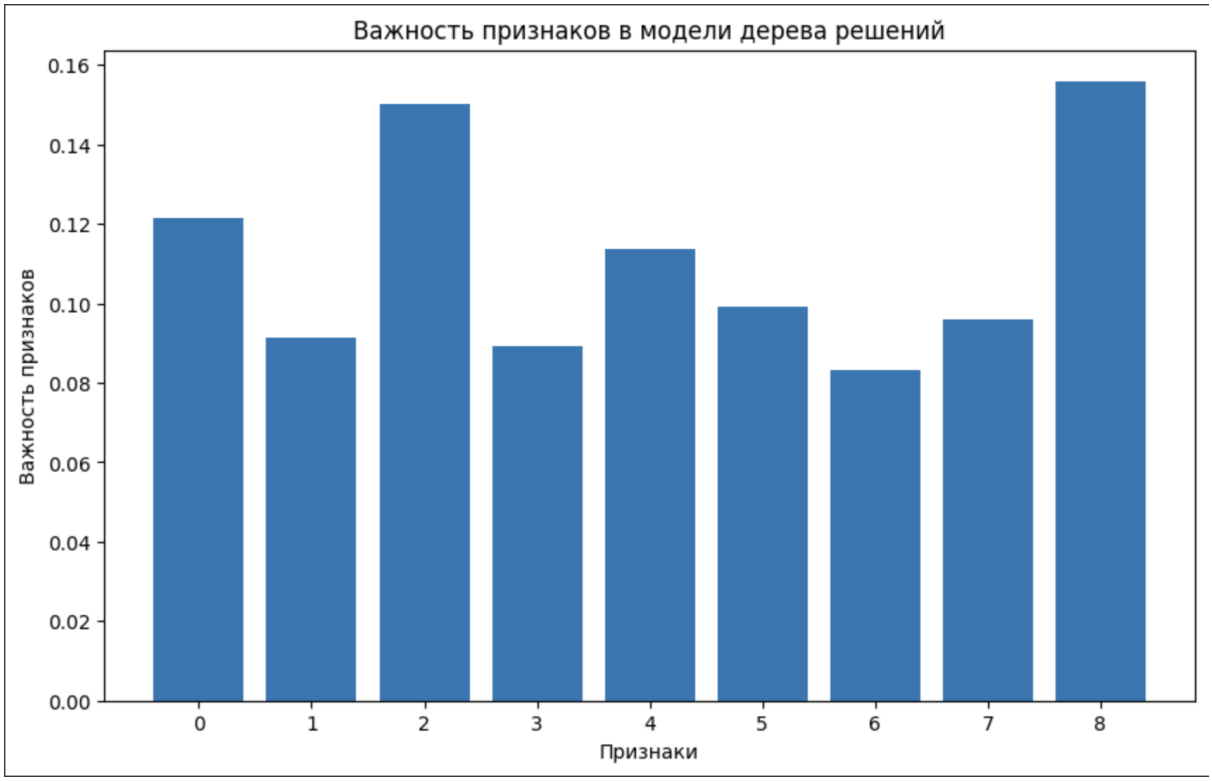

Рисунок 18 - Важность признаков в случайном лесу

Опираясь на полученный график, можно выделить два наиважнейших признака:

- X3 – число браков на 1000 человек
- X9 – число зарегистрированных преступлений на 100000 населения

Исходя из полученных результатов, можно утверждать, что модель дерева решений не подходит для нашей задачи, так как из всех признаков она использует только один, в то время как модель случайного леса и логистическая модель используют все признаки, но в разных соотношениях.

Вывод

В ходе выполнения лабораторной работы были изучены 4 различных модели и методы их работы: KNN, логистическая регрессия, дерево решений и случайный лес. Также было изучено, как определить качество полученных моделей и провести сравнение между ними.