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 15

Название:

Дисциплина: Прикладной анализ данных

Студент ИУ6-55Б _____ В.К. Полубояров
(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.

Исходя из варианта были подготовлены исходные данные - отобраны строки, соответствующие регионам: Центральный, Центрально-Черноземный, Северо-Кавказский, Западно-Сибирский, Восточно-Сибирский, Дальневосточный районы.

С помощью библиотеки Pandas были получены данные в виде массивов, где X - целевые признаки, Y - целевые значения.

Для разделения на тестовую и обучающую выборку воспользуемся методом train_test_split.

Зададим random_seed для возможности воспроизведения полученных результатов.

Для приблизительно одинакового разбиения по классам воспользуемся стратификацией классов с помощью одноименного аргумента stratify = y.

```
df = pd.read_excel('data_2.xlsx')
df = df.drop(df.columns[[0, 1]], axis=1)
X = df.drop(['Ybin'], axis = 1)
y = df['Ybin']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size=0.85,
                                                    random_state=42,
                                                    stratify=y)
print(f"Количество строк в y_train по классам: {np.bincount(y_train)}")
print(f"Количество строк в y_test по классам: {np.bincount(y_test)}")
```

```
Количество строк в y_train по классам: [28 13]
Количество строк в y_test по классам: [6 2]
```

Рисунок 1 - разбиение данных на две выборки.

Как мы видим, отношение (0 к 1) в тестовой выборке равно 3, а в обучающей 2,15, что можно считать приемлемым распределением таргета в двух выборках.

Задание 2. Нормирование данных

Для корректной работы с данными необходимо нормировать данные. Для нормирования была выбрана “Стандартизация”, которую часто называют Z-оценкой. Она рассчитывается с помощью формулы, отдельно для каждого x.

Для корректной работы посчитаем mean() и std() на основе обучающих данных. Затем, используя полученные значения, нормируем обучающие и тестовые данные.

```
mean_X = X_train.mean()
std_X = X_train.std()

#нормализация тренировочных и тестовых данных
for column in X_train.columns:
    X_train[column] = (X_train[column] - mean_X[column]) / std_X[column]
    X_test[column] = (X_test[column] - mean_X[column]) / std_X[column]
```

Рисунок 2 - нормирование данных.

Задание 3. Fit-predict модели kNN с перебором гиперпараметров по сетке.

Для подбора гиперпараметра создадим сетку, где k будет перебираться от 2 до 32. Для перебора параметров воспользуемся методом GridSearchCV, зададим:

- KNN - метод обучения
- Нашу сетку с гипермараметрами
- Метод оценки модели - по точности
- Кросс-валидация - 5 (принятое дефолтное значение)

После обучения модели с подобранным гиперпараметром на тренировочных данных построим предсказания.

```
knn = KNeighborsClassifier()
#создадим сетку параметров
grid_space={'n_neighbors': range(2,33,1)}
grid = GridSearchCV(knn, param_grid=grid_space, scoring='accuracy', cv=5)
knn_model = grid.fit(X_train, y_train)

#Делаем предсказания
knn_predictions = knn_model.predict(X_test)

print(classification_report(y_test, knn_predictions))
```

```
              precision    recall  f1-score   support

           0       0.86      1.00      0.92         6
           1       1.00      0.50      0.67         2

    accuracy                           0.88         8
   macro avg       0.93      0.75      0.79         8
weighted avg       0.89      0.88      0.86         8
```

Рисунок 3 - KNN

Задание 4. Fit-predict модели логистической регрессии с подбором гиперпараметров по сетке.

По заданию необходимо подобрать метод регуляризации с помощью сетки. Чтобы понять, какие методы нам доступны, надо определиться с методом решения (solver).

solver : *{'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'}, default='lbfgs'*

Algorithm to use in the optimization problem. Default is 'lbfgs'. To choose a solver, you might want to consider the following aspects:

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones;
- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss;
- 'liblinear' is limited to one-versus-rest schemes.
- 'newton-cholesky' is a good choice for `n_samples` >> `n_features`, especially with one-hot encoded categorical features with rare categories. Note that it is limited to binary classification and the one-versus-rest reduction for multiclass classification. Be aware that the memory usage of this solver has a quadratic dependency on `n_features` because it explicitly computes the Hessian matrix.

Рисунок 4 - solver

Воспользуемся методом решения 'liblinear', так как у нас небольшое количество данных.

Warning: The choice of the algorithm depends on the penalty chosen. Supported penalties by solver:

- 'lbfgs' - ['l2', None]
- 'liblinear' - ['l1', 'l2']
- 'newton-cg' - ['l2', None]
- 'newton-cholesky' - ['l2', None]
- 'sag' - ['l2', None]
- 'saga' - ['elasticnet', 'l1', 'l2', None]

Рисунок 5 - доступные методы регуляризации

Следовательно, нам необходимо определить используем мы L1 или L2.

```
logistic = LogisticRegression(solver='liblinear')
#создадим сетку параметров
grid_space={'penalty': ['l2', 'l1']}
grid = GridSearchCV(logistic, param_grid=grid_space, cv=5)
logistic_model = grid.fit(X_train, y_train)
print(grid.best_params_)
#Делаем предсказания
logistic_predictions = logistic_model.predict(X_test)

print(classification_report(y_test, logistic_predictions))
```

```
{'penalty': 'l2'}
              precision    recall  f1-score   support

           0       1.00      0.83      0.91         6
           1       0.67      1.00      0.80         2

    accuracy                           0.88         8
   macro avg       0.83      0.92      0.85         8
weighted avg       0.92      0.88      0.88         8
```

Рисунок 6 - Логистическая регрессия.

В ходе подбора был выбран L2, как параметр регуляризации.

Задание 5. Fit-predict модели дерева решений с подбором гиперпараметров по сетке.

По заданию необходимо подобрать параметр глубины дерева с помощью сетки. Зададим random_state = 42 для воспроизводимости результатов.

```
decision_tree = DecisionTreeClassifier(splitter ='best',random_state=RANDOM_SEED)
grid_space={'max_depth': range (1, 100, 1)}
grid = GridSearchCV(decision_tree, param_grid=grid_space, cv=5)
decision_tree_model = grid.fit(X_train, y_train)
print(grid.best_params_)
#Делаем предсказания
decision_tree_predictions = decision_tree_model.predict(X_test)

print(classification_report(y_test, decision_tree_predictions))
```

```
{'max_depth': 2}
              precision    recall  f1-score   support

           0       0.83      0.83      0.83         6
           1       0.50      0.50      0.50         2

    accuracy                           0.75         8
   macro avg       0.67      0.67      0.67         8
weighted avg       0.75      0.75      0.75         8
```

Рисунок 7 - дерево решений.

В ходе работы алгоритма по перебору было выбрано значение глубины равное двум.

Теперь обучим модель случайного леса. Также зададим параметр random_state = 42.

```
forest = RandomForestClassifier(random_state=RANDOM_SEED)
grid_space={'max_depth': range (1, 20, 1)}
grid = GridSearchCV(forest, param_grid=grid_space, cv=5)
forest_model = grid.fit(X_train, y_train)
print(grid.best_params_)
#Делаем предсказания
forest_predictions = forest_model.predict(X_test)

print(classification_report(y_test, forest_predictions))
```

```
{'max_depth': 1}
              precision    recall  f1-score   support

           0       0.83      0.83      0.83         6
           1       0.50      0.50      0.50         2

    accuracy                           0.75         8
   macro avg       0.67      0.67      0.67         8
weighted avg       0.75      0.75      0.75         8
```

Рисунок 8 - случайный лес.

В ходе работы алгоритма максимальная глубина была выбрана равной 1.

Задание 6-7. Сравнение качества моделей.

Посчитаем порог отсечения (threshold) для каждой из модели. Для этого будем для каждого порога считать F1-меру, которая “балансирует” показатели precision и recall. Чем выше значение F1, тем сбалансированнее precision и recall между собой.

Построим графики зависимости F1(threshold) для каждой из моделей и определим пороги отсечения.

Код будет одинаковым для каждой из модели, отличие будет только в типе модели. Код представлен на рисунке 9.

```
thresholds = np.linspace(0, 1, 1000)
f1_scores = []
y_scores = knn_model.predict_proba(X_test)[:,1]
for threshold in thresholds:
    y_pred = [1 if score >= threshold else 0 for score in y_scores]
    f1 = f1_score(y_test, y_pred)
    f1_scores.append(f1)

plt.figure(figsize=(10, 6))
plt.plot(thresholds, f1_scores)
plt.xlabel('Threshold')
plt.ylabel('F1 Score')
plt.grid(True)
plt.title('F1 Score vs. Threshold')
plt.xticks(np.arange(0, 1.1, 0.05))
plt.show()
```

Рисунок 9 - построение графика F1(threshold)

Получим график для модели KNN. Возьмем threshold = 0.55.

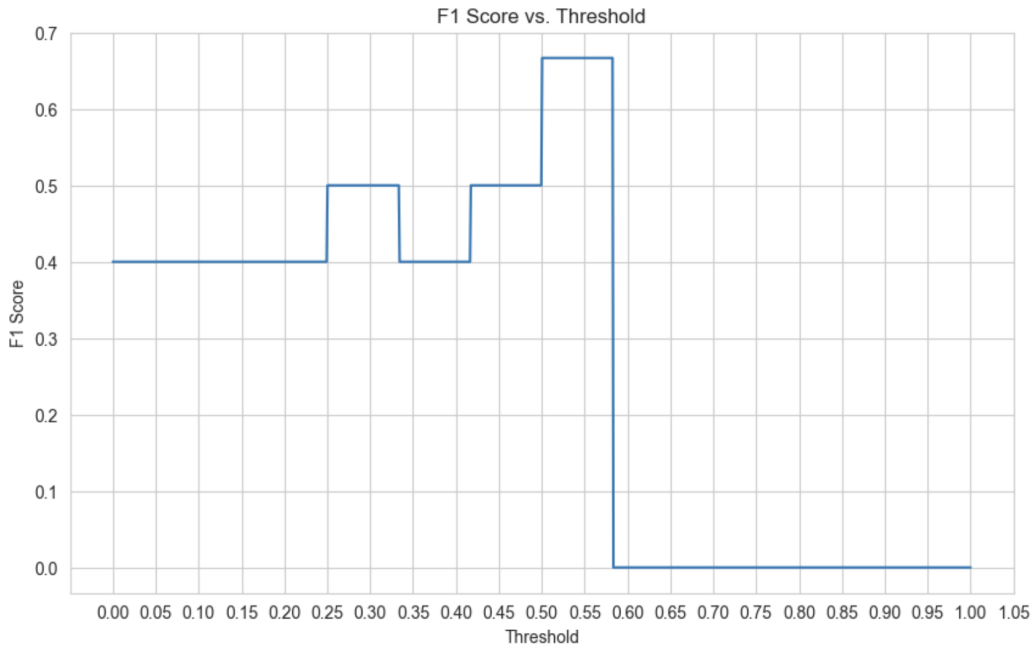


Рисунок 10 - F1 для KNN.

Получим график для логистической регрессии. Возьмем threshold = 0.55.

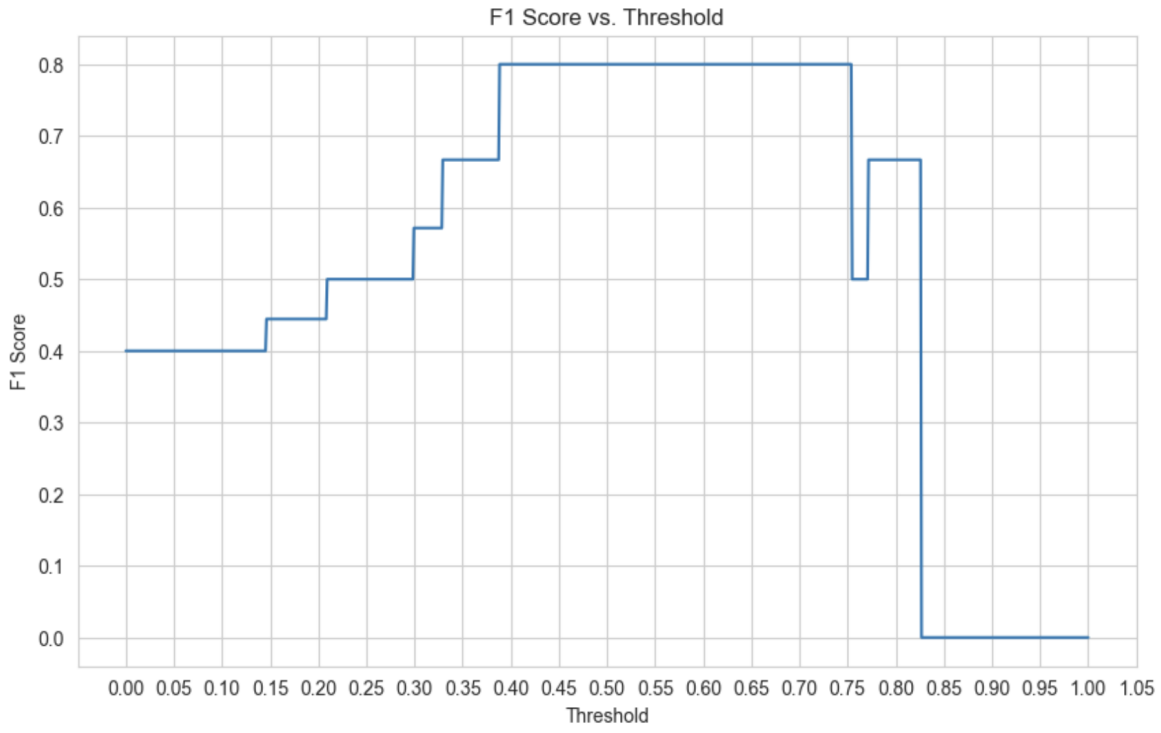
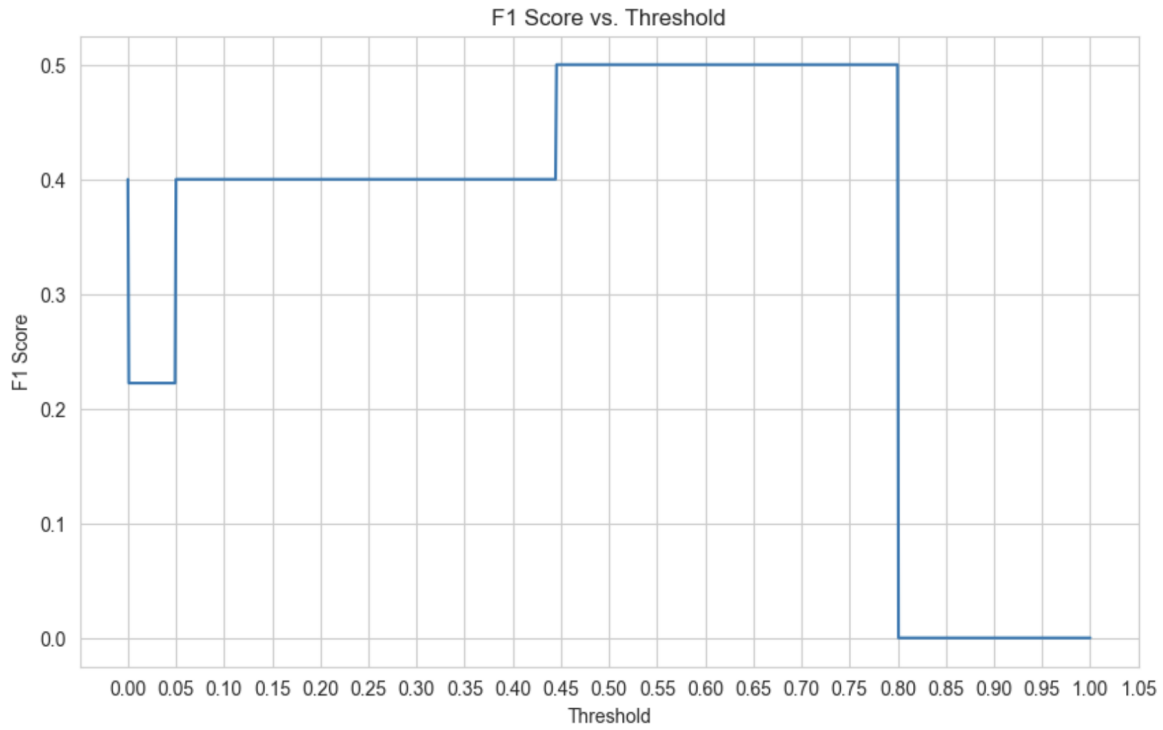


Рисунок 11 - F1 для логистической регрессии.

Получим график для дерева решений. Возьмем threshold = 0.55.



Получим график для случайного леса. Возьмем threshold = 0.3.

Рисунок 12 - F1 для дерева решения.

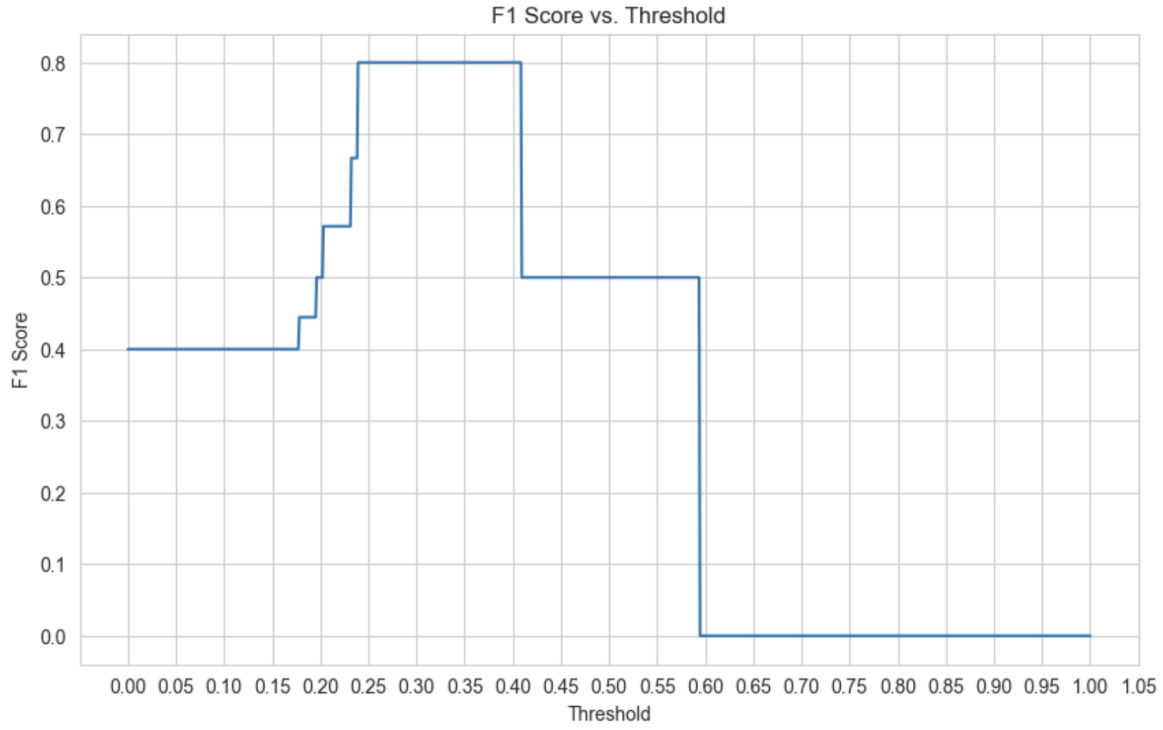


Рисунок 13 - F1 для случайного леса.

Построим таблицу полученных метрик для каждой из модели на **тестовых** данных.

	KNN	ЛГ	ДР	СЛ
Accuracy	0.875	0.875	0.75	0.875
Precision	1.0	0.66	0.5	0.66
Recall	0.5	1.0	0.5	1.0
F1	0.66	0.8	0.5	0.8
ROC-AUC	0.75	0.9166	0.66	0.916

Accuracy - это метрика, которая характеризует качество модели, агрегированное по всем классам.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Она показывает как хорошо наша модель угадывает классы. Однако, эта метрика в нашем случае крайне нерепрезентативна, так как у нас в в тестовой выборке в несколько раз больше 0, чем 1. Это значит, если наша модель будет всегда выдавать 0, то показатель точности будет достаточно высоким, хотя предсказать 1 мы не сможем никогда.

Рассмотрим precision. Данная метрика показывает долю правильно предсказанных положительных объектов среди всех объектов, предсказанных положительным классом. Precision показывает способность отличать класс от других классов.

$$\text{Precision} = \frac{TP}{TP + FP}$$

У KNN самый высокий precision, затем идут ЛГ и СЛ, затем ДР.

Recall (полнота) показывает долю правильно найденных положительных объектов среди всех объектов положительного класса. Recall демонстрирует способность алгоритма обнаруживать данный класс в целом.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Самый высокий recall у ЛГ и СЛ, затем KNN и ДР.

Таким образом, KNN одновременно является лучшей в плане отличительной способности, но при этом худшая с точки зрения полноты.

ЛГ и СЛ получили одинаковые показатели с идеальным recall и чуть выше среднего precision.

ДР оказался самым неоптимальным алгоритмом, обладающим посредственным precision и recall.

F_1-мера — среднее гармоническое между precision и recall. Самый лучший F1 у ЛГ и СЛ. Затем KNN. Самый худший показатель - ДР.

Введем две метрики

TPR (true positive rate) – это полнота, доля положительных объектов, правильно предсказанных положительными:

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

FPR (false positive rate) – это доля отрицательных объектов, неправильно предсказанных положительными:

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

Рассмотрим ROC-AUC (receiver operating characteristic) (area under the curve). ROC-AUC равен доле пар объектов вида (объект класса 1, объект класса 0), которые алгоритм верно упорядочил. Чем ближе его значение к 1, тем идеальнее наша модель.

Самый высокий показать этой метрики у ЛГ и СЛ. Затем идет KNN. Самый худший результат - ДР.

Исходя из полученных результатов по всем метрикам можно сделать вывод, что для наших данных:

- 1 место - ЛГ и СЛ
- 2 место - KNN
- 3 место - ДР

Теперь посчитаем все те же метрики на основе обучающих данных.

	KNN	ЛГ	ДР	СЛ
Accuracy	0.707	0.78	0.829	0.780
Precision	1.0	0.83	0.8	0.642
Recall	0.0769	0.384	0.615	0.6923
F1	0.1428	0.526	0.695	0.666
ROC-AUC	0.538	0.674	0.7719	0.7568

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.

	KNN	ЛГ	ДР	СЛ
Accuracy	↑	↑	↓	↑
Precision	=	↓	↓	↑
Recall	↑	↑	↓	↑
F1	↑	↑	↓	↑
ROC-AUC	↑	↑	↓	↑

Исходя из полученных результатов можно сделать вывод, что проблема переобучения явно выражена в дереве решений, так как показатели всех метрик на тестовых данных сильно ниже, чем на обучающих.

В остальных моделях все показатели выросли, кроме показателя Precision в модели логистической регрессии, который лишь немного уменьшился (на 0.2).

Задание 8. Важность признаков.

В модели логистической регрессии важность признаков не оценивается напрямую, как в древесных моделях. Вместо этого, веса коэффициентов при признаках указывают на их важность. Чем больше по модулю коэффициент при признаке, тем больший вклад вносит этот признак в прогнозы модели. Положительные коэффициенты могут указывать на положительное влияние признака, а отрицательные - на отрицательное влияние. Посмотрим на веса нашей модели.

	0	1	2	3	4	5	6	7	8
0	0.655183	-0.143715	0.619507	-1.042959	-0.197644	-0.600343	-0.662546	-0.761243	0.425402

Рисунок 14 - веса логистической регрессии.

Наиболее значимыми признаками стали:

- X4 - число разводов на 1000 человек
- X8 - численность населения с денежными доходами ниже прожиточного минимума в % от численности населения
- X7 - соотношение средней оплаты труда и прожиточного минимума трудоспособного населения
- X1 - рождаемость населения на 1000 человек

В деревьях решений важность признаков можно оценить с помощью атрибута feature_importances_ после обучения модели. Этот атрибут предоставляет значение важности каждого признака в модели. Чем выше значение, тем важнее признак. Интерпретация важности признаков в деревьях решений может быть более сложной, но обычно она основана на том, как часто признак используется для разделения данных и насколько хорошо разделение по этому признаку улучшает качество прогнозов.

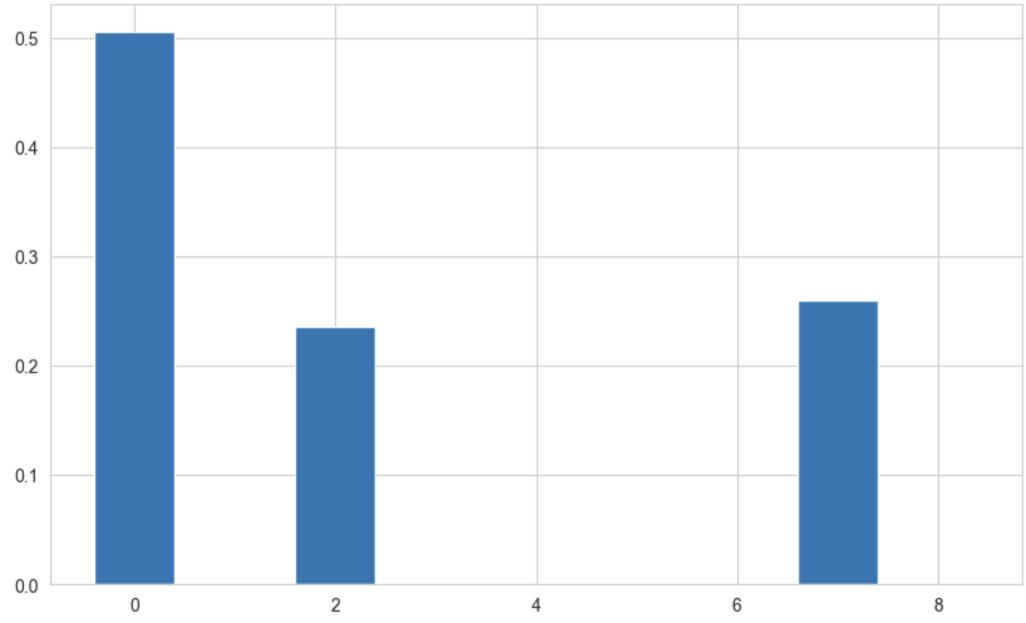


Рисунок 15 - важность признаков в дереве решений

В результате получилось три важных признака:

- X1 - рождаемость населения на 1000 человек
- X3 - число браков на 1000 человек
- X7 - соотношение средней оплаты труда и прожиточного минимума трудоспособного населения

В случайном лесе важность признаков также может быть оценена с помощью атрибута feature_importances_, и это обычно делается путем усреднения важности признаков по всем деревьям в лесу. Эта оценка дает более стабильную и надежную оценку важности признаков по сравнению с одиночным деревом. Интерпретация важности признаков в случайном лесе аналогична интерпретации важности в деревьях решений, но с учетом усреднения результатов по всем деревьям.

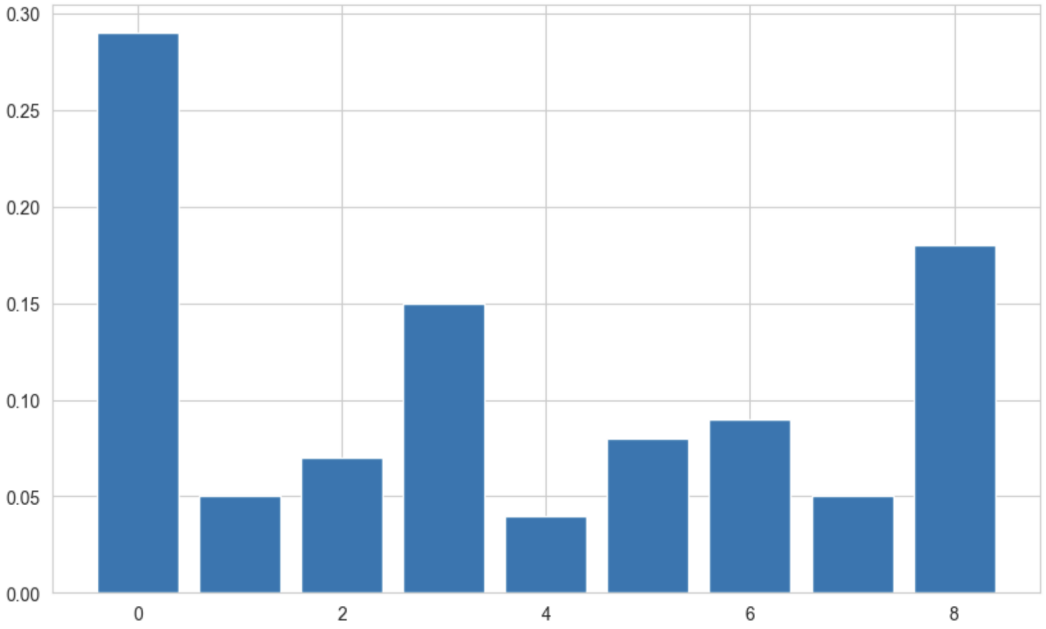


Рисунок 16 - важность признаков в случайном лесу

В результате самые важные три признака:

- X1 - рождаемость населения на 1000 человек
- X8 - численность населения с денежными доходами ниже прожиточного минимума в % от численности населения
- X3 - число браков на 1000 человек

Во всех трех моделях присутствует X1 (рождаемость населения на 1000 человек) как один из самых значимых параметров.

X8 присутствует в ЛГ и СЛ.

X3 присутствует в ДР и СЛ.

Исходя из полученных результатов, можно сказать что дерево решений не очень удачный метод для нашей задачи, так как из всех признаков он использует только три признака, в то время как СЛ и ЛГ использует все признаки в той или иной степени.

Можно предположить, что плохие показатели дерева решений связаны как раз с тем, что модель использует только 3 из 8 признаков для вынесения предсказаний.

Вывод

В ходе выполнения лабораторной работы были изучены 4 различных модели и методы их работы: KNN, логистическая регрессия, дерево решений и случайный лес. Также были изучены основные метрики, применяемые для определения качества моделей.