Министерство науки и высшего образования РФ

Томский государственный университет
систем управления и радиоэлектроники

С. А. Давыденко, С. А. Фоминых, Е. К. Сенчихин, Е.Ю. Костюченко

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В МЕДИЦИНСКОЙ ДИАГНОСТИКЕ

Методические указания к лабораторным работам
для студентов направлений подготовки
09.04.04 «Программная инженерия»

УДК 004.8

ББК 32.813.5

Д 13

Давыденко, С. А. ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В МЕДИЦИНСКОЙ ДИАГНОСТИКЕ: Учебно-методическое пособие С. А. Давыденко, Е. К. Сенчихин, С. А. Фоминых, Е.Ю. Костюченко. — Томск: ТУСУР, 2024. — 73 с.

Настоящее учебно-методическое пособие содержит описания практических работ по дисциплине «Искусственный интеллект в медицинской реабилитации» для направлений подготовки, входящих в укрупненную группу специальностей и направлений 09.04.04 «Программная инженерия».

Одобрено на заседании кафедры КИБЭВС протокол №7 от 30.08.2024 года

УДК 004.8

ББК 32.813.5

Содержание

ЛАБОРАТОРНАЯ РАБОТА №1

Прогнозирование деменции на основе медицинских данных пациентов

Целью работы является прогнозирование деменции на основе медицинских данных пациентов.

Краткие теоретические сведения

Деменция — это синдром, характеризующийся снижением когнитивных функций, таких как память, мышление, ориентация, понимание и способность к выполнению повседневных задач.

Для прогнозирования деменции используется набор данных с информацией о пациентах, включая их возраст, пол, количество визитов к врачу, уровень образования и т.д.

Random Forest Classifier — это ансамблевый метод машинного обучения, использующий множество деревьев решений. Он подходит для решения задач классификации и регрессии.

Логистическая регрессия - это разновидность множественной регрессии, общее назначение которой состоит в анализе связи между несколькими независимыми переменными (называемыми также регрессорами или предикторами) и зависимой переменной.

SVM — это метод машинного обучения, который ищет гиперплоскость, разделяющую классы данных с максимальным отступом. В этом коде используется стратегия "One-vs-Rest" для многоклассовой классификации и построение ROC-кривой для оценки качества классификации.

KNN — это простой алгоритм, который классифицирует объекты на основе их ближайших соседей. Алгоритм основывается на гипотезе, что похожие объекты находятся рядом в пространстве признаков. В этом коде также используется стратегия "One-vs-Rest" и построение ROC-кривой.

One-vs-Rest — стратегия для многоклассовой классификации, которая превращает многоклассовую задачу в несколько бинарных задач, обучая отдельную модель для каждого класса.

ROC-кривая (Receiver Operating Characteristic) — это график, который показывает соотношение между чувствительностью (True Positive Rate) и специфичностью (False Positive Rate) при разных порогах классификации. ROC-кривая помогает оценить качество классификации, особенно в случае дисбаланса классов.

AUC (Area Under the Curve) — метрика, показывающая площадь под ROC-кривой. Чем ближе значение AUC к 1, тем лучше модель отделяет классы.

Предобработка данных включает удаление ненужных столбцов, преобразование категориальных данных в числовые, стандартизацию признаков и разделение на обучающую и тестовую выборки.

1 Ход работы

1.1 Загрузка данных

Перед началом работы необходимо скачать набор данных Dementia Prediction Dataset. Он представляет собой таблицу, содержащую данные пациентов.

Ссылка на набор данных:

https://www.kaggle.com/datasets/shashwatwork/dementia-prediction-dataset

Ссылка на Google colab:

https://colab.research.google.com/drive/1YbFjoZp63p-CrpKZYXB8CBcOZXMmpsLB?usp=sharing

После того как набор данных скачан, необходимо установить библиотеки и указать путь к набору данных.

```
# Загрузка данных
file_path = 'path_to_your_dataset.csv'  # Укажите путь к вашему датасету
data = pd.read_csv(file_path)
```

1.2 Подготовка данных

Далее необходимо подготовить данные, удалить лишние столбцы, преобразовать некоторое данные в числовые, разделить данные на признаки, сделать обучающие и тестовые выборки.

Удаляются столбцы, которые не содержат полезной информации для прогнозирования деменции. Например:

Subject ID: идентификатор пациента, который не несет информацию для прогнозирования.

MRI ID: идентификатор МРТ, также не содержит информации для прогнозирования.

```
# Удаление ненужных столбцов
data = data.drop(['Subkect ID', 'MRI ID'], axis=1)
```

Категориальные переменные преобразуются в числовые значения, чтобы их можно было использовать в модели машинного обучения. Это необходимо, так как большинство алгоритмов машинного обучения работает с числовыми данными.

Group: преобразуем значения "demented", "nondemented", "converted" в числа (например: 0, 1, 2).

M/F: преобразуем пол ("M", "F") в числа (например: 0, 1).

Hand: преобразуем руку (правая "R", левая "L") в числа (например: 0, 1).

```
# Преобразование категориальных данных в числовые
label_encoder_group = LabelEncoder()
data['Group'] = label_encoder_group.fit_transform(data['Group'])
```

Приведённый ниже код выполняет стандартизацию данных. Стандартизация преобразует данные таким образом, что они будут иметь нулевое среднее значение и единичное стандартное отклонение.

```
# Стандартизация признаков
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

1.3 Обучение модели

На этапе обучения каждый классификатор обучается на тренировочных данных. При этом происходит настройка весов или других внутренних параметров модели в зависимости от алгоритма. Например:

- Логистическая регрессия подбирает коэффициенты для линейной комбинации признаков.
- SVM находит гиперплоскость, которая максимально разделяет классы.
- Метод K-ближайших соседей определяет, какие точки использовать для классификации новых данных.
- Дерево решений строит правила для разделения данных на основе значений признаков.
- Случайный лес создает ансамбль из деревьев решений, каждое из которых обучается на случайной подвыборке данных.
- Градиентный бустинг обучает деревья последовательно, уменьшая ошибки предыдущих моделей.
- MLP обучается методом обратного распространения ошибки, настраивая веса, чтобы минимизировать функцию потерь.

1.4 Оценка модели

Для того чтобы выполнить оценку качества работы модели после получения предсказаний (занесены в переменную y_pred), используйте следующие функции:

- confusion_matrix: выводит матрицу запутанностей, которая показывает количество правильных и неправильных предсказаний по каждому классу.
- classification_report: выводит такие метрики, как точность (precision), полнота (recall), F1-мера и т.д.

Помимо этого примените метрику accuracy_score, которая вычисляет общую точность модели (доля правильных предсказаний). Ожидаемый формат вывода результатов показан на рисунке 1.1.

```
#Оценка модели
y_pred = model.predict(X_test)
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))
# Матрица запутанностей
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d')
plt.title("Матрица запутанностей")
plt.show()
```

```
Confusion Matrix:
[[ 3  1  7]
 [ 1 31  0]
 [ 0  1 31]]

Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.27      0.40        11
           1       0.94      0.97      0.95        32
           2       0.82      0.97      0.89        32

    accuracy                           0.87        75
   macro avg       0.84      0.74      0.75        75
weighted avg       0.86      0.87      0.84        75


Accuracy Score:
0.8666666666666667
```

Рисунок 1.1 – Оценка качества модели

Постройте ROC-кривую, пример для которой приведён на рисунке 1.2. График ROC представляет собой зависимость чувствительности (True Positive Rate, TPR) от ложноположительного уровня (False Positive Rate, FPR) при различных пороговых значениях вероятности для классификатора. Для расчёта графиков ROC-кривой используются следующие метрики:

- True Positive Rate (TPR): Доля положительных примеров, правильно классифицированных моделью как положительные (для нас это пациенты с деменцией, правильно предсказанные как такие).
- False Positive Rate (FPR): Доля отрицательных примеров, которые классифицируются как положительные (пациенты без деменции, ошибочно предсказанные как больные).

Площадь под ROC-кривой (AUC, Area Under the Curve) — это числовое выражение качества модели. AUC варьируется от 0 до 1:

- AUC = 1: Идеальная модель, которая верно отличает классы при любом пороге.
- AUC = 0.5: Модель не отличается от случайного угадывания.
- AUC < 0.5: Модель хуже случайного угадывания (редкий случай в качественной модели).

Чем ближе AUC к 1, тем лучше модель. В случае с прогнозированием деменции AUC поможет определить, насколько точно модель предсказывает наличие заболевания по результатам теста или медицинских данных.

```
# Прогнозирование вероятностей
y_pred_proba = model.predict_proba(X_test_scaled)[:, 1]
# Оценка модели через AUC-ROC
roc_auc = roc_auc_score(y_test, y_pred_proba)
print(f"ROC AUC: {roc_auc:.2f}")
# Построение ROC-кривой
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC-кривая')
plt.legend()
plt.show()
```

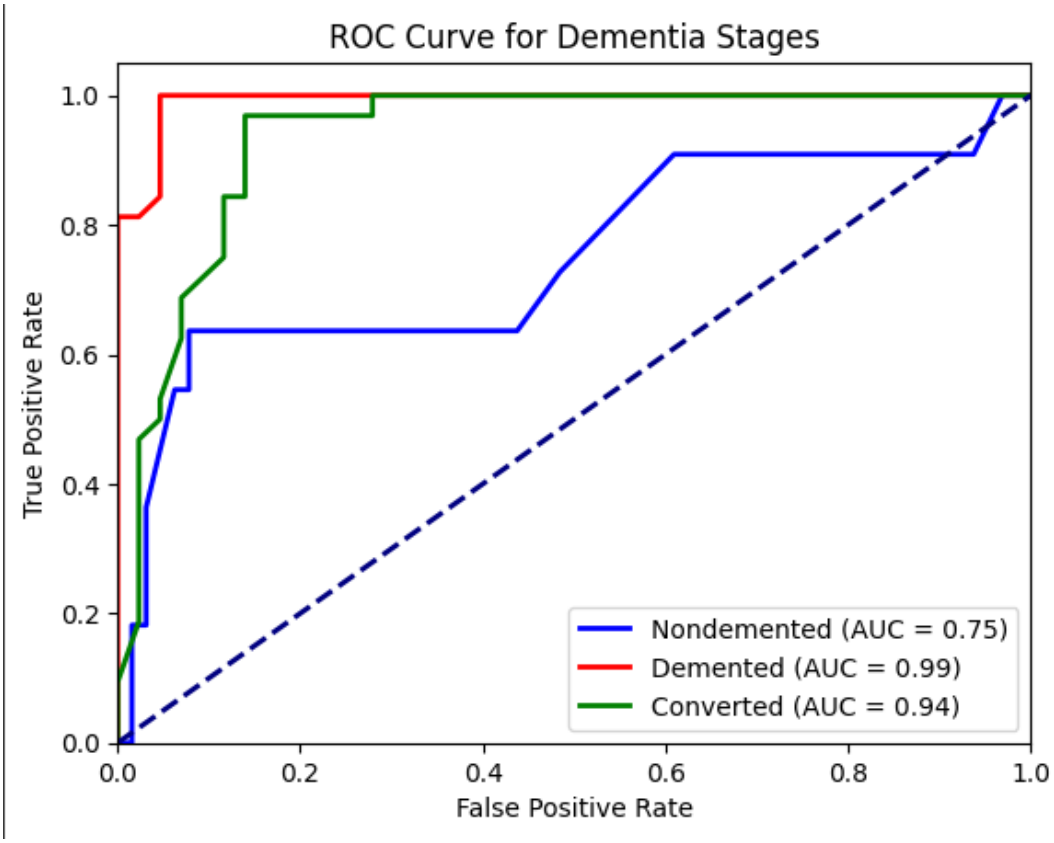


Рисунок 1. 2 – ROC-кривая

2 Задание на лабораторную работу

После выполнения работы с первым набором данных, сохранить и протестировать на других наборах данных, сравнить результаты. В таблице 2.1 представлены варианты индивидуального задания.

Таблица 2.1 – Индивидуальное задание

Вар.	Набор данных	Метод обучения модели	Оценка модели
1	https://www.kaggle.com/code/lordxerxes/dementia-prediction/input	Random Forest Classifier	Матрица
2	https://www.kaggle.com/code/gkitchen/predicting-dementia/input	Логистическая регрессия	ROC-кривая
3	https://www.kaggle.com/code/lordxerxes/dementia-prediction/input	SVM	Матрица
4	https://www.kaggle.com/code/gkitchen/predicting-dementia/input	KNN	ROC-кривая

Контрольные вопросы

1. Какие бывают методы обучения модели и чем они отличаются?
2. Что такое One-vs-Rest и для чего используется?
3. Что такое ROC-кривая?
4. Что показывает AUC (Area Under the Curve)?
5. Для чего нужна предобработка данных?

ЛАБОРАТОРНАЯ РАБОТА №2

Обнаружение признаков болезни сердца с использованием методов машинного обучения

Целью работы является обнаружение признаков болезни сердца с использованием методов машинного обучения.

Краткие теоретические сведения

Ансамбль — это метод машинного обучения, который использует несколько моделей для повышения точности предсказаний. Вместо того, чтобы полагаться на одну модель, ансамблевые методы объединяют результаты нескольких базовых моделей для создания более устойчивого и точного предсказания. Эти базовые модели могут быть однородными (например, несколько решающих деревьев в случайном лесу) или разнородными (например, объединение деревьев решений и моделей опорных векторов). Ансамбли работают по принципу, что группа слабых моделей может превзойти одну мощную модель.

Bagging (Bootstrap Aggregating) - это метод, который улучшает точность модели, объединяя несколько базовых моделей, обученных на различных подвыборках данных.

Boosting - методы, которые уменьшают смещение, последовательно обучая модели, каждая из которых пытается улучшить ошибки предыдущей.

Stacking - ансамбль, который обучает мета-модель, комбинируя выходы нескольких базовых моделей.

VotingClassifier - объединяет несколько моделей, где финальное предсказание делается на основе голосования (медиана для регрессии или мажоритарное голосование для классификации).

1 Ход работы

1.1 Загрузка данных

Перед началом работы необходимо скачать набор данных ECG Arrhythmia Classification Dataset.

Ссылка на набор данных:

https://www.kaggle.com/datasets/sadmansakib7/ecg-arrhythmia-classification-dataset?select=Sudden+Cardiac+Death+Holter+Database.csv

Ссылка на Google colab:

https://colab.research.google.com/drive/1SINBQTF6-AMFqeWZKFGHHYw9wK9NZxht?usp=sharing

После того как набор данных скачан, необходимо установить библиотеки и указать путь к набору данных.

1.2 Подготовка данных

Далее необходимо подготовить данные, удалить лишние столбцы, преобразовать некоторое данные в числовые, разделить данные на признаки, сделать выборки.

Данные загружаются в виде таблицы с использованием библиотеки pandas. На этом этапе важно получить первое представление о содержимом набора данных. Просмотрите структуру набора данных — столбцы, типы данных, количество строк и наличие пропусков. Это помогает понять, с какими признаками предстоит работать.

Используйте isnull() или info() для проверки столбцов на наличие пропусков. Если пропуски редкие, строки могут быть удалены. Для числовых данных пропуски можно заменить на среднее, медиану или моду (чаще всего встречающееся значение) признака. Для категориальных признаков пропуски могут быть заполнены наиболее частым значением или меткой "Unknown", если это оправдано.

Пример кода с заменой на медиану:

```
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Cholesterol'].fillna(df['Cholesterol'].mean(), inplace=True)
```

Выбросы (значения, значительно отклоняющиеся от других данных) могут негативно влиять на обучение модели. Для их поиска можно использовать визуализации, такие как ящики с усами (boxplots), либо считать значения, находящиеся вне определённых квартильных диапазонов (например, за пределами 1.5 IQR).

1.5 IQR — это метод статистического анализа для выявления выбросов, основанный на интерквартильном размахе (IQR). Интерквартильный размах показывает диапазон, в котором лежат средние 50% данных, и вычисляется как разность между третьим квартилем (Q3, верхний квартиль) и первым квартилем (Q1, нижний квартиль): IQR=Q3−Q1

Обработка выбросов выполняется следующими способами:

- Удаление выбросов: можно удалить строки с экстремальными значениями (это данные, которые значительно отличаются от остальных наблюдений в наборе данных), если это оправдано.
- Корректировка: в некоторых случаях данные можно скорректировать, заменив выбросы на граничные значения.

Модели машинного обучения не работают с текстовыми данными, поэтому категориальные переменные нужно преобразовать в числовые. Методы кодирования категориальных признаков:

- One-Hot Encoding: Для признаков с немногими категориями (например, пол, наличие или отсутствие симптома) используют побитное кодирование с помощью функции pd.get_dummies().
- Label Encoding: Если категорий много или они имеют порядок (например, уровни холестерина), используется LabelEncoder, который назначает каждой категории уникальное число.

Пример кодирования категориальных признаков:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

```
df['Gender'] = le.fit_transform(df['Gender'])
df = pd.get_dummies(df, columns=['ChestPainType', 'RestingECG'])
```

1.3 Обучение модели

Выполните обучение моделей и ансамблей.

```
# Bagging с базовой моделью (например, решающее дерево)
bagging = BaggingClassifier(n_estimators=100, random_state=42)
bagging.fit(X_train_scaled, y_train)
# Прогнозирование
y_pred_bagging = bagging.predict(X_test_scaled)
```

Постройте ROC-кривую и матрицу запутанностей (рисунки 1.1-1.2).

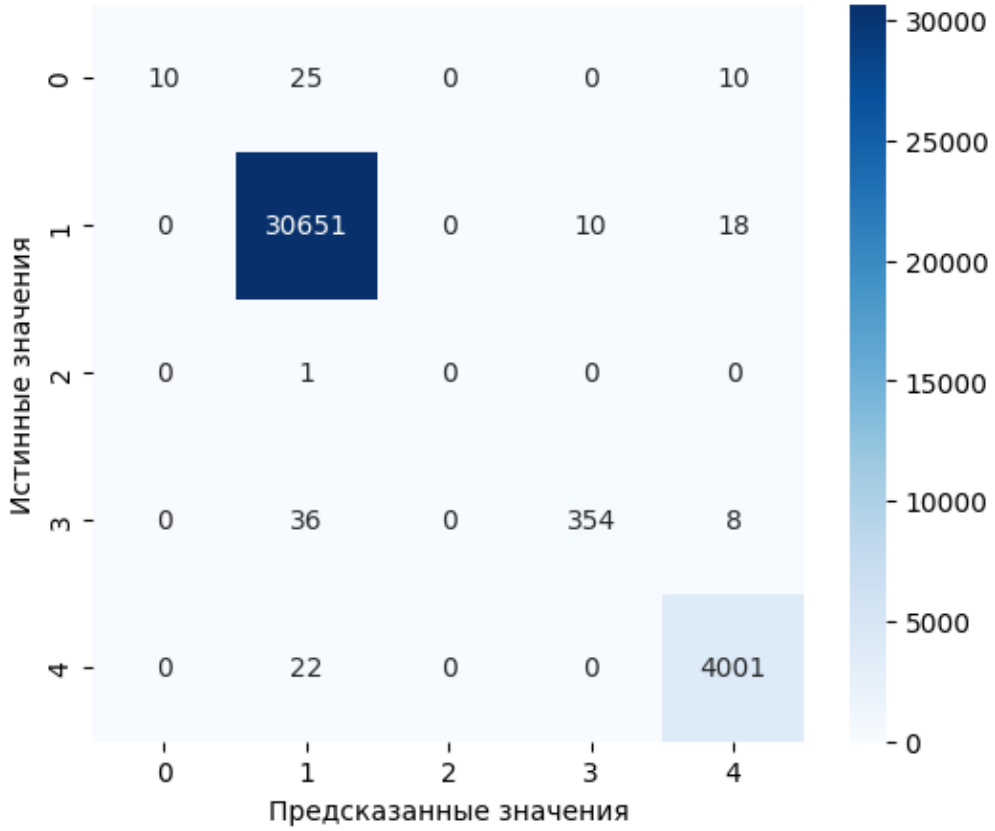


Рисунок 1.1 – Матрица запутанностей

Матрица запутанностей представляет собой квадратную таблицу с четырьмя значениями:

True Positives (TP): Количество правильных предсказаний класса VEB.

True Negatives (TN): Количество правильных предсказаний класса N.

False Positives (FP): Количество ошибочных предсказаний, когда модель предсказала VEB, но истинный класс был N.

False Negatives (FN): Количество ошибочных предсказаний, когда модель предсказала N, но истинный класс был VEB.

В данной задаче матрица запутанностей помогает увидеть:

- Частоту ошибок модели: Например, сколько раз модель ошибочно классифицировала N как VEB и наоборот.
- Баланс между классами: Если ошибки распределены неравномерно, это может говорить о возможной необходимости балансировки классов или настройки параметров модели.

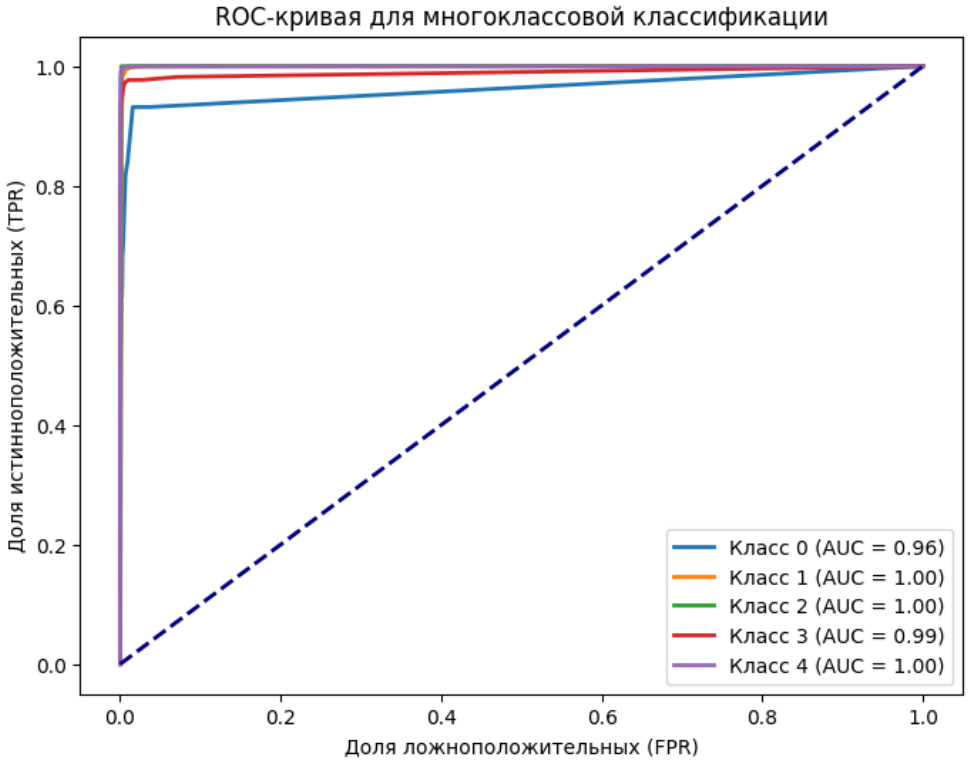


Рисунок 1.2 – Пример ROC-кривой

Каждый класс представлен отдельной кривой:

- График демонстрирует, насколько хорошо модель отделяет положительные примеры одного класса от всех остальных.
- В легенде указаны классы и их значения AUC (Area Under Curve).

Что означают FPR и TPR:

- FPR (False Positive Rate): доля объектов, неверно отнесённых к текущему классу (ложноположительные предсказания).
- TPR (True Positive Rate): доля объектов, правильно предсказанных для текущего класса (истинно положительные предсказания).

Площадь под кривой (AUC):

- Значение AUC показывает, насколько хорошо модель различает данный класс. Чем ближе AUC к 1, тем лучше модель справляется.

В данном случае:

- Класс 0 имеет AUC = 0.96.
- Классы 1, 2, 3 и 4 имеют AUC = 1.00 или близко к этому, что свидетельствует о высокой точности модели для этих классов.

Диагональная линия:

- Синяя пунктирная линия показывает поведение случайной модели (без обучения). Если кривая близка к этой линии, модель работает плохо.

Итог:

- Модель демонстрирует высокую способность различать классы, особенно для классов с AUC = 1.00.
- Для класса 0 качество чуть ниже, но всё равно достаточно высокое (AUC = 0.96).

2 Задание на лабораторную работу

Выполните задание по вариантам, используя тот же набор данных. В таблице 2.1 представлены варианты индивидуального задания

Таблица 2.1 – Индивидуальное задание

Вариант	**Ансамбль**
1	Boosting
2	Stacking
3	AdaBoost
4	Gradient Boosting
5	VotingClassifier

Контрольные вопросы

1. Что такое ансамбль и для чего он нужен?
2. Что такое Bagging?
3. Что такое Stacking и чем отличается от Bagging?
4. Когда используется Boosting?
5. Чем отличается VotingClassifier от Bootstrap Aggregating?

ЛАБОРАТОРНАЯ РАБОТА №3

Разработка модели ИИ для диагностики заболеваний легких с помощью рентгенографии

Целью работы является разработка модели ИИ для диагностики заболеваний легких с помощью рентгенографии.

Краткие теоретические сведения

Пневмония, или воспаление лёгких (Pneumonia) — вариант острой респираторной инфекции, поражающей лёгочную ткань. Лёгкие состоят из небольших мешотчатых образований (альвеол), которые в ходе акта дыхания здорового человека должны наполняться воздухом. При пневмонии альвеолы заполнены жидкостью (экссудатом) и гноем, которые ухудшают газообмен.

CNN (Convolutional Neural Network, сверточная нейронная сеть) — это один из типов нейронных сетей, специально разработанный для работы с изображениями, хотя он также может использоваться для других типов данных, таких как видео, аудиофайлы и даже тексты.

Сверточные слои (Convolutional Layers): В этих слоях используются фильтры, которые "сканируют" изображение, извлекая пространственные особенности, такие как края, текстуры и другие признаки. Это позволяет модели распознавать сложные паттерны на изображениях.

Слои подвыборки (Pooling Layers): Этот слой уменьшает размер данных, сохраняя важные признаки. Это помогает уменьшить вычислительную нагрузку и защищает модель от переобучения.

Полносвязные слои (Fully Connected Layers): Эти слои принимают выходы сверточных слоев и преобразуют их в конечный результат, например, вероятности классов.

Многослойный перцептрон (MLP, Multilayer Perceptron) - Это базовый вид искусственных нейронных сетей, состоящий из нескольких

полносвязных слоев (без сверточных слоев). Модель учится на основе данных, переданных в виде плоского вектора.

RNN — это архитектура нейронных сетей, которая особенно хорошо подходит для работы с последовательностями данных, где есть зависимость между текущим и предыдущими шагами (например, временные ряды, тексты).

1 Ход работы

1.1 Загрузка и предварительная обработка данных

Перед началом работы необходимо скачать набор данных Chest X-Ray Images (Pneumonia). Он представляет собой набор фотографий с рентгеном здоровых и больных людей.

Ссылка на набор данных:

https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

Ссылка на Google colab:

https://colab.research.google.com/drive/1jd154VpkbIS1BAMWTNLkt8F4O6v2hQQm?usp=sharing

Данные загружаются с помощью ImageDataGenerator, который также выполняет масштабирование и аугментацию изображений.

Генератор тренировочных данных используется для подачи данных в модель во время обучения. Валидационные данные не участвуют в процессе обучения. Их цель — дать представление о том, насколько хорошо модель обобщает информацию, полученную при обучении, на неизвестных данные. Этот набор используется для промежуточной проверки модели на каждом этапе обучения. Генератор тестовых данных используется исключительно для оценки окончательной производительности модели, уже обученной и настроенной. Тестовые данные представляют собой абсолютно неизвестные для модели изображения.

Тренировочный генератор: помогает модели учиться на максимальном объёме данных, сохраняя разнообразие.

Валидационный генератор: контролирует обобщающую способность модели и помогает избежать переобучения, не включаясь в обучение.

Тестовый генератор: предоставляет финальную метрику качества модели в условиях, приближенных к её реальному использованию

1.2 Создание CNN модели

Свёрточная нейронная сеть (CNN) — это архитектура нейронной сети, специально разработанная для анализа визуальных данных, таких как изображения. Она способна выявлять различные признаки (или паттерны), которые помогают модели различать классы (в данном случае, здоровые лёгкие и лёгкие с пневмонией).

Основные компоненты CNN:

- Входной слой: принимает изображения размером, заданным пользователем, например, 150×150 пикселей.
- Свёрточные слои (Conv2D): эти слои применяют фильтры для сканирования изображения и выделения простых признаков, например, краёв, текстур и форм. С каждым слоем сеть начинает распознавать более сложные признаки, например, структуры, которые могут быть характерны для пневмонии. Пример: Conv2D(filters=32, kernel_size=(3, 3), activation='relu') — свёрточный слой с 32 фильтрами размером 3×3, применяет функцию активации ReLU.
- Слои подвыборки (Pooling Layers): эти слои уменьшают размерность данных (снижают разрешение изображения), чтобы сократить количество вычислений и выявить более значимые признаки. Обычно используется MaxPooling2D(pool_size=(2, 2)), который выбирает максимальное значение в каждом 2×2 блоке.
- Полносвязные слои (Dense Layers): после того как признаки выделены, используется один или несколько полносвязных слоёв для окончательной классификации. Обычно перед полносвязными слоями данные

"выпрямляются" (Flatten()), чтобы превратить двумерные данные в одномерный вектор.

- Выходной слой: это последний слой, определяющий итоговый класс. Для задачи бинарной классификации (нормальные/пневмония) выходной слой имеет 1 нейрон с сигмоидной функцией активации (activation='sigmoid'). Для многоклассовой классификации использовалась бы функция softmax.

Пример создания CNN модели для диагностики заболеваний лёгких приведён ниже.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
# Создаём модель
model = Sequential()
# Первый свёрточный блок
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(MaxPooling2D(pool_size=(2, 2)))
…
# Переход к полносвязному слою
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))  # Для предотвращения переобучения
# Выходной слой
model.add(Dense(1, activation='sigmoid'))
```

Dropout(0.5) добавляется для регуляризации, чтобы снизить переобучение, временно «отключая» 50% нейронов в слое на каждой эпохе.

Компиляция модели с оптимизатором adam, который помогает автоматически регулировать скорость обучения, и binary_crossentropy как функцией потерь, которая наиболее подходит для бинарной классификации.

1.3 Обучение модели

На этапе обучения модель настраивает свои параметры на тренировочном наборе данных, чтобы научиться отличать классы. Процесс обучения включает в себя несколько важных шагов.

Подготовка генераторов данных: создайте три генератора данных для подачи изображений в модель (тренировочный, валидационный и тестовый).

Генератор для тренировочных данных включает аугментацию (случайные повороты, сдвиги, отражения), что помогает модели обобщить признаки.

```
train_generator = train_datagen.flow_from_directory(
    'path_to_train_data',
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)
val_generator = val_datagen.flow_from_directory(
    'path_to_validation_data',
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)
```

Параметры обучения:

- Эпохи: Определяют количество полных проходов по тренировочным данным. Увеличение числа эпох позволяет модели лучше изучить данные, однако это может привести к риску переобучения.
- Пакеты (Batch Size): Количество изображений, используемых для обновления параметров сети на каждом шаге. Более высокие значения пакетов ускоряют обучение, но требуют больше памяти.

Функции обратного вызова (Callback):

- Early Stopping: Завершает обучение, если точность на валидационном наборе перестаёт улучшаться.
- TensorBoard: Визуализирует процесс обучения, графики функции потерь и точности, а также метрики по эпохам.

Запуск обучения:

Модель обучается на тренировочных данных, корректируя свои параметры (веса) для минимизации ошибки на каждом шаге. После каждой эпохи модель оценивается на валидационном наборе, чтобы оценить её обобщающую способность.

В коде представленном ниже реализуется процесс обучения модели с использованием генераторов данных и заданными функциями обратного вызова, которые помогают улучшить и оптимизировать обучение.

```
history = model.fit(
    train_generator,
    epochs=50,
    validation_data=val_generator,
    callbacks=[earlystop_cb, tensorboard_callback]
```

1.4 Оценка модели на тестовом наборе и визуализация результатов обучения

Модель проходит через весь тестовый набор данных, предсказывая классы для каждого изображения. Эти предсказания сравниваются с истинными метками классов. На основе этих предсказаний рассчитываются основные метрики качества модели, такие как:

Точность (accuracy) — это метрика, отражающая долю правильных предсказаний модели относительно общего числа предсказаний.

Функция потерь (loss) на тестовом наборе показывает степень ошибки модели при классификации тестовых данных, оценивая, насколько её предсказания отклоняются от истинных значений.

Тестовая точность показывает, какой процент тестовых изображений модель классифицировала правильно. Тестовая функция потерь показывает, насколько сильно предсказанные значения отклоняются от реальных меток в среднем.

Постройте графики визуализации динамики функции потерь (рисунок 1.1) и точности (рисунок 1.2).

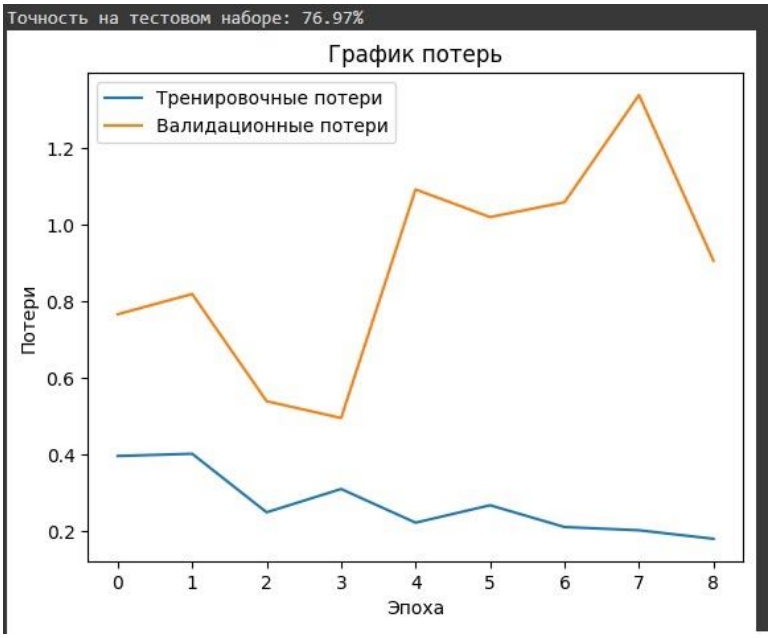


Рисунок 1.1 – График потерь

Этот график показывает, насколько хорошо или плохо модель предсказывает правильные результаты. На графике отображаются:

Training loss (тренировочные потери): показатель ошибки модели, рассчитанный на тренировочных данных.

Validation loss (валидационные потери): мера ошибки модели, определяемая на валидационных данных, используемых для проверки её обобщающей способности.

Если валидационные потери начинают расти при дальнейшем снижении тренировочных потерь, это признак переобучения (overfitting), когда модель слишком хорошо запоминает тренировочные данные, но не обобщает на новые данные.

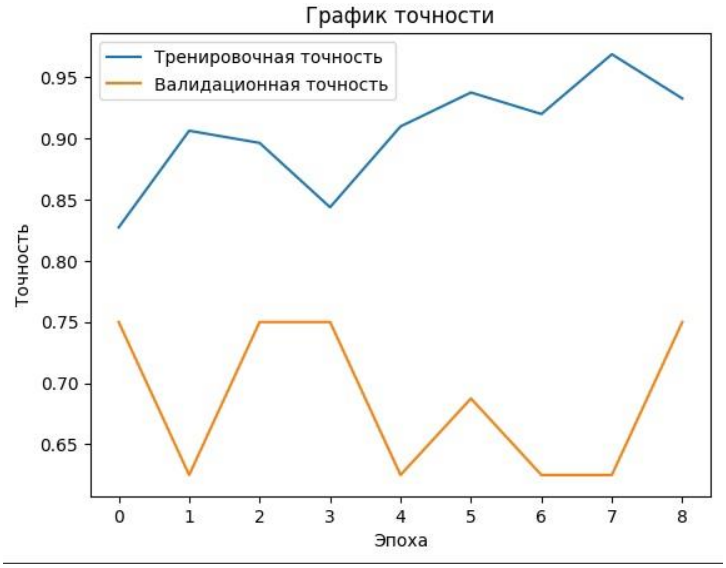


Рисунок 1.2 – График точности

График точности показывает процент правильных предсказаний модели.

На графике отображаются:

Training accuracy (Точность на тренировочных данных): процент правильных предсказаний на тренировочном наборе.

Validation accuracy (Точность на валидационных данных): процент правильных предсказаний на валидационном наборе.

2 Задание на лабораторную работу

После выполнения всех пунктов методического указания, выполните задание по вариантам с тем же набором данных.

Используйте последние две цифры номера студенческого билета, в качестве процентов от общего числа данных, которые будут использоваться при выполнении задания по варианту (например, номер студенческого билета ******79, а вариант 1, то будет использоваться модель MLP и обучаться она будет используя 79% данных).

Сравнить полученные результаты с результатами работы моделей по варианту. В таблице 2.1 представлены варианты индивидуального задания

Таблица 2.1 – Индивидуальное задание

Вариант	**Модели**
1	MLP, KNN
2	RNN, MLP
3	SVM, RNN
4	KNN, SVM

Контрольные вопросы

1. Что такое пневмония?
2. Что такое сверточная нейронная сеть?
3. Какие бывают слои?
4. Что такое MLP?
5. Что такое RNN?

ЛАБОРАТОРНАЯ РАБОТА №4

Разработка модели ИИ для выявления депрессии на основе данных пациентов

Целью работы является разработка модели ИИ для выявления депрессии на основе результатов опроса людей.

1 Ход работы

1.1 Загрузка данных

Перед началом работы необходимо скачать набор данныхThe RHMCD-20 datasets for Depression and Mental Health Data Analysis with Machine Learning. Он представляет собой таблицу, в которой содержатся результаты опроса людей.

Далее необходимо импортировать необходимые библиотеки, загрузить файл в DataFrame (рисунок 1.1)

	Age	Gender	Occupation	Days_Indoors	Growing_Stress	Quarantine_Frustrations	Changes_Habits	Mental_Health_History	Weight_Change	Mood_Swings	Coping_Struggles	Work_Interest	Social_Weakness
0	20-25	Female	Corporate	1-14 days	Yes	Yes	No	Yes	Yes	Medium	No	No	Yes
1	30-Above	Male	Others	31-60 days	Yes	Yes	Maybe	No	No	High	No	No	Yes
2	30-Above	Female	Student	Go out Every day	No	No	Yes	No	No	Medium	Yes	Maybe	No
3	25-30	Male	Others	1-14 days	Yes	No	Maybe	No	Maybe	Medium	No	Maybe	Yes
4	16-20	Female	Student	More than 2 months	Yes	Yes	Yes	No	Yes	Medium	Yes	Maybe	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...
819	20-25	Male	Corporate	Go out Every day	No	Yes	No	Yes	Yes	Medium	No	Yes	Maybe
820	20-25	Male	Others	1-14 days	Yes	Yes	No	Yes	Maybe	Low	No	Maybe	Maybe
821	20-25	Male	Student	More than 2 months	Yes	Maybe	Maybe	No	Yes	High	Yes	Yes	Maybe
822	16-20	Male	Business	15-30 days	No	No	Maybe	No	Maybe	Low	Yes	No	Maybe
823	30-Above	Female	Others	15-30 days	No	No	No	No	Yes	Low	Yes	No	Maybe

Рисунок 1.1 – Загруженные в DataFrame данные

Далее выполните визуализацию данных: sns.countplot создает график, показывающий распределение значений в столбце Growing_Stress, который указывает на уровень стресса у участников (рисунок 1.2)

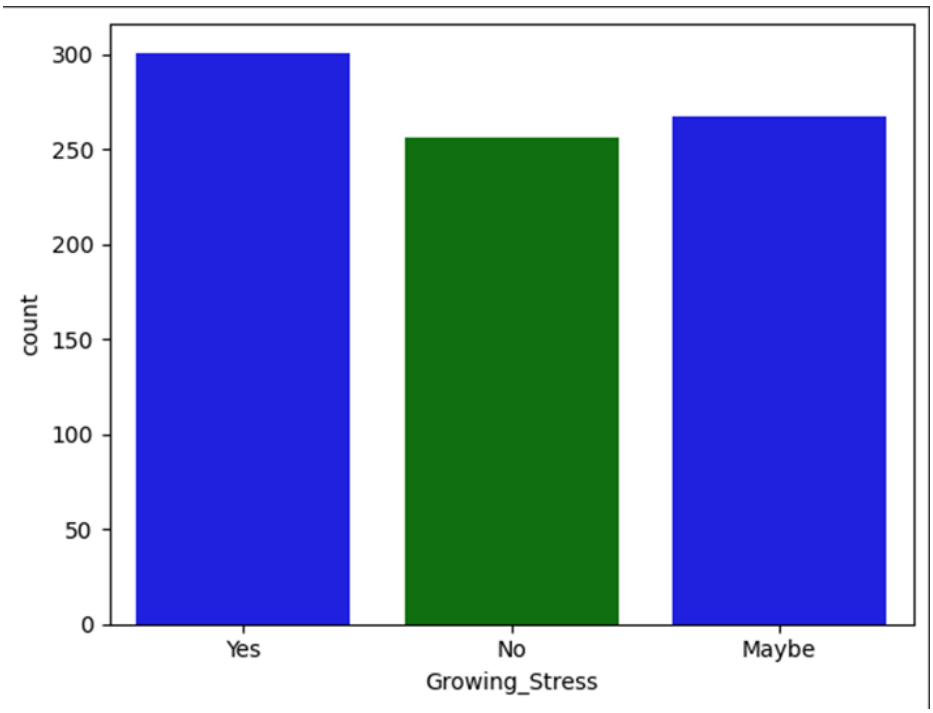


Рисунок 1.2 – Визуализация данных

1.2 Обработка и разделение данных

Для преобразования категориальных переменных в числовые необходимо применить функцию LabelEncoder. На рисунке 1.3 представлен результат преобразования.

	Age	Gender	Occupation	Days_Indoors	Growing_Stress	Quarantine_Frustrations	Changes_Habits	Mental_Health_History	Weight_Change	Mood_Swings	Coping_Struggles	Work_Interest	Social_Weakness
0	1	0	1	0	2	2	1	2	2	2	0	1	2
1	3	1	3	2	2	2	0	1	1	0	0	1	2
2	3	0	4	3	1	1	2	1	1	2	1	0	1
3	2	1	3	0	2	1	0	1	0	2	0	0	2
4	0	0	4	4	2	2	2	1	2	2	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
819	1	1	1	3	1	2	1	2	2	2	0	2	0
820	1	1	3	0	2	2	1	2	0	1	0	0	0
821	1	1	4	4	2	0	0	1	2	0	1	2	0
822	0	1	0	1	1	1	0	1	0	1	1	1	0
823	3	0	3	1	1	1	1	1	2	1	1	1	0

Рисунок 1.3 – Преобразованные данные

X содержит все признаки, кроме целевой переменной *y*, которая представляет Growing_Stress.

Разделите данные на тренировочный и тестовый наборы с помощью train_test_split, чтобы оценить качество модели на данных, которые она не видела во время обучения.

1.3 Создание и обучение модели

Для создания базовой модели воспользуйтесь классом Sequential. Настройте простую архитектуру, начав с одного скрытого слоя, содержащего 32 нейрона и функцию активации sigmoid. Затем добавьте второй скрытый слой с 16 нейронами и функцией активации relu, чтобы модель могла захватывать более сложные зависимости в данных. Завершите модель выходным слоем из 3 нейронов с функцией активации softmax, что обеспечит многоклассовую классификацию (по числу категорий в целевой переменной Growing_Stress) (рисунок 1.4). Модель компилируется с использованием функции потерь sparse_categorical_crossentropy (подходит для задач многоклассовой классификации) и оптимизатора Adam.

Эта конфигурация является лишь отправной точкой. Подумайте о дальнейшем улучшении модели, добавляя больше слоев или изменяя функции активации и количество нейронов, чтобы повысить её точность и устойчивость.

```
# и создадим первый скрытый слой (с указанием функции активации и размера входного слоя)
model.add(Dense(32, activation = 'sigmoid'))

# затем второй скрытый слой
model.add(Dense(16, activation = 'relu'))

# и наконец выходной слой
model.add(Dense(3, activation = 'softmax'))
```

Рисунок 1.4 – Фрагмент кода со слоями

Для улучшения процесса обучения и контроля его качества в модели используются несколько специальных методов — TensorBoard, EarlyStopping и ReduceLROnPlateau. Эти компоненты задаются при компиляции и запуске модели через параметр callbacks, и каждый из них играет ключевую роль в стабильности и эффективности обучения:

- TensorBoard используется для детальной визуализации процесса обучения, позволяя отслеживать такие параметры, как изменения точности и потерь в каждой эпохе.

- EarlyStopping автоматически останавливает обучение, если потери на валидационном наборе перестают улучшаться на протяжении заданного количества эпох. Это помогает предотвратить избыточное обучение и сохранить оптимальную версию модели.
- ReduceLROnPlateau уменьшает скорость обучения (learning rate), если валидационные потери остаются стабильными и не снижаются. Это позволяет модели обучаться более плавно, когда она выходит на плато, что способствует лучшему поиску минимальных значений функции потерь.

Модель обучается на тренировочных данных в течение 500 эпох с использованием заданных функций обратного вызова (рисунок 1.5).

```
Epoch 1/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.3667 - loss: 1.1015 - learning_rate: 0.0010
Epoch 2/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3726 - loss: 1.0944 - learning_rate: 0.0010
Epoch 3/500
 1/21 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.4375 - loss: 1.1136/usr/local/lib/python3.10
  current = self.get_monitor_value(logs)
/usr/local/lib/python3.10/dist-packages/keras/src/callbacks/callback_list.py:96: UserWarning: Lear
  callback.on_epoch_end(epoch, logs)
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3860 - loss: 1.0946 - learning_rate: 0.0010
Epoch 4/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3593 - loss: 1.0908 - learning_rate: 0.0010
Epoch 5/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.3924 - loss: 1.0888 - learning_rate: 0.0010
Epoch 6/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.3777 - loss: 1.0864 - learning_rate: 0.0010
Epoch 7/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.4145 - loss: 1.0779 - learning_rate: 0.0010
Epoch 8/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.3744 - loss: 1.0870 - learning_rate: 0.0010
Epoch 9/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.4163 - loss: 1.0838 - learning_rate: 0.0010
Epoch 10/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3858 - loss: 1.0834 - learning_rate: 0.0010
Epoch 11/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4295 - loss: 1.0835 - learning_rate: 0.0010
Epoch 12/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.3885 - loss: 1.0956 - learning_rate: 0.0010
Epoch 13/500
21/21 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4330 - loss: 1.0772 - learning_rate: 0.0010
```

Рисунок 1.5 – Обучение модели

1.4 Оценка модели

После завершения обучения модели важно оценить её производительность на независимом тестовом наборе данных. Для этого используется метод model.evaluate. Этот метод позволяет получить объективную метрику точности модели и функцию потерь на данных, которые

не использовались в процессе тренировки. Такая оценка даёт представление о том, насколько хорошо модель обобщает знания и справляется с задачей классификации на реальных данных.

Вызов model.evaluate запускает процесс, в ходе которого модель делает прогнозы по тестовому набору и сравнивает их с реальными значениями, рассчитывая итоговые показатели (рисунок 1.6). Эти позволяют судить, достаточно ли эффективна базовая модель.

```
model.evaluate(
  X_test,
  y_test
)
```

```
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2888 - loss: 1.2736
[1.269910454750061, 0.2969697117805481]
```

Рисунок 1.6 – Оценка модели

Далее с помощью функции accuracy_score вычислите точность модели на тестовых данных. Выведите отчет о классификации с использованием classification_report, который включает метрики, такие как precision, recall и F1-score (рисунок 1.7).

```
WARNING:tensorflow:5 out of the last 13 calls to <funct
6/6 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Точность модели: 29.70%
Отчет о классификации:
              precision    recall  f1-score   support

           0       0.34      0.34      0.34        62
           1       0.25      0.17      0.20        52
           2       0.28      0.37      0.32        51

    accuracy                           0.30       165
   macro avg       0.29      0.29      0.29       165
weighted avg       0.29      0.30      0.29       165
```

Рисунок 1.7 – Оценка точности модели

1.5 Построение матрицы запутанностей

Функия confusion_matrix позволяет создать матрицу запутанностей, показывающую, как предсказания модели сопоставляются с истинными значениями.

На рисунке 1.8 представлена визуализация матрицы запутанностей с использованием библиотеки seaborn. График отображает, насколько хорошо модель классифицирует каждую категорию.

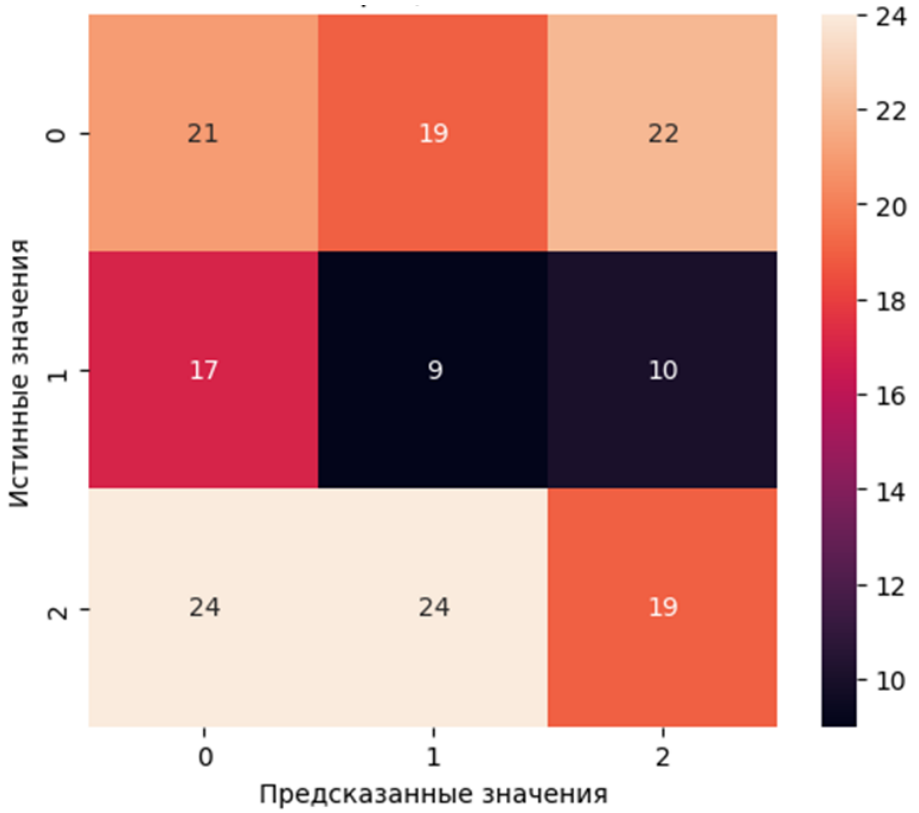


Рисунок 1.8 – Матрица запутанностей

2 Задание на лабораторную работу

Изучить набор данных и предоставленный блокнот Google colab. Выполнить кодирование категориальных данных. Несколько раз изменить архитектуру модели и обучить ее, полученные результаты свести в таблицу и сделать выводы о зависимости изменения архитектуры и значения метрик.

Контрольные вопросы

1. Что такое депрессия?
2. Каким образом происходит кодирование переменных?
3. Что такое полносвязный слой?
4. Какие параметры существуют у слоя Dense?
5. Для чего используется параметр callbacks при обучении модели?

ЛАБОРАТОРНАЯ РАБОТА № 5

Использование искусственного интеллекта для определения сахарного диабета

Целью работы является использование искусственного интеллекта для определения сахарного диабета.

Краткие теоретические сведения

Са́харный диабе́т — группа эндокринных заболеваний, связанных с нарушением усвоения глюкозы вследствие абсолютной или относительной недостаточности гормона инсулина.

Одним из основных способов борьбы с несбалансированными данными является увеличение (oversampling) и уменьшение (undersampling) выборки. Эти методы направлены на достижение баланса между классами путем изменения количества примеров в каждом классе.

1 Ход работы

Перед началом работы необходимо создать копию блокнота https://colab.research.google.com/drive/1sIOnFJU0_SRQ77GRF0p1LgKSaw3mHkrP?usp=sharing.

Далее необходимо импортировать библиотеки необходимые для работы.

```
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, recall_score, f1_score
from sklearn.svm import SVC
```

Следующим шагом следует загрузить набор данных в среду Colab, затем импортировать данные из CSV-файла в переменную df и отобразить первые несколько строк.(рисунок 1.1).

```
df = pd.read_csv('/content/diabetes.csv')
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Рисунок 1.1 – Просмотр первых записей таблицы

Для получения названий столбцов используется команда «columns» (рисунок 1.2).

```
df.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

Рисунок 1.2 – Название столбцов

Для получения общей информации о наборе данных существует команда «info», она позволяет узнать тип каждого признака, а также есть ли в данных пропуски (рисунок 1.3).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Рисунок 1.3 – Общая информация о наборе данных

Далее необходимо вывести матрицу корреляции для просмотра зависимости между атрибутами (рисунок 1.4).

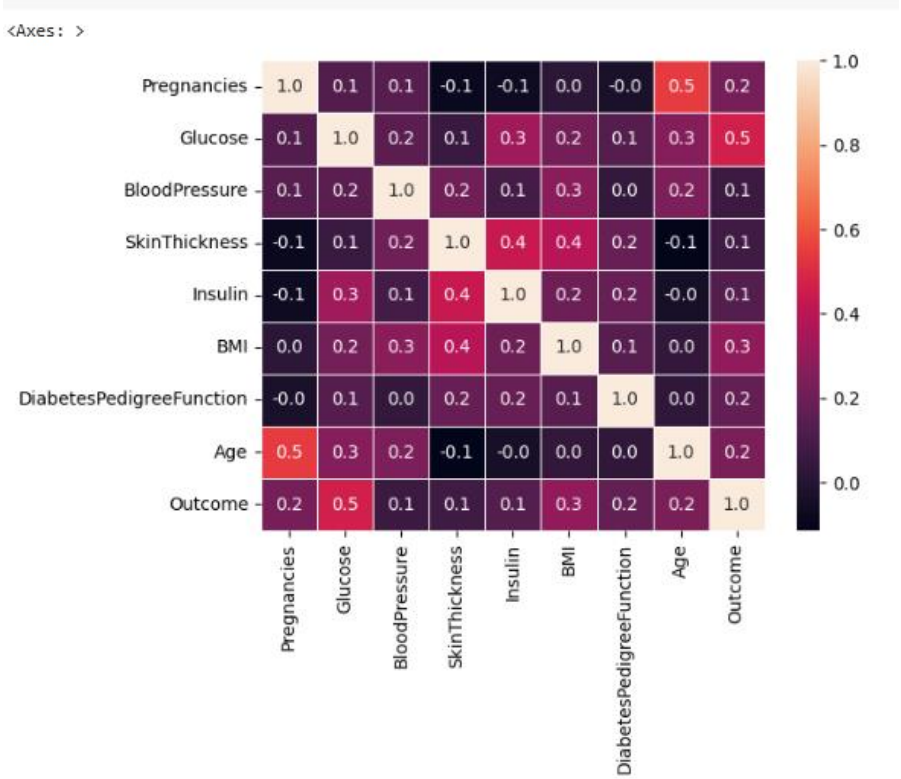


Рисунок 1.4 – Матрица корреляции

Распределение по классам можно посмотреть с помощью команды «countplot» (рисунок 1.5).

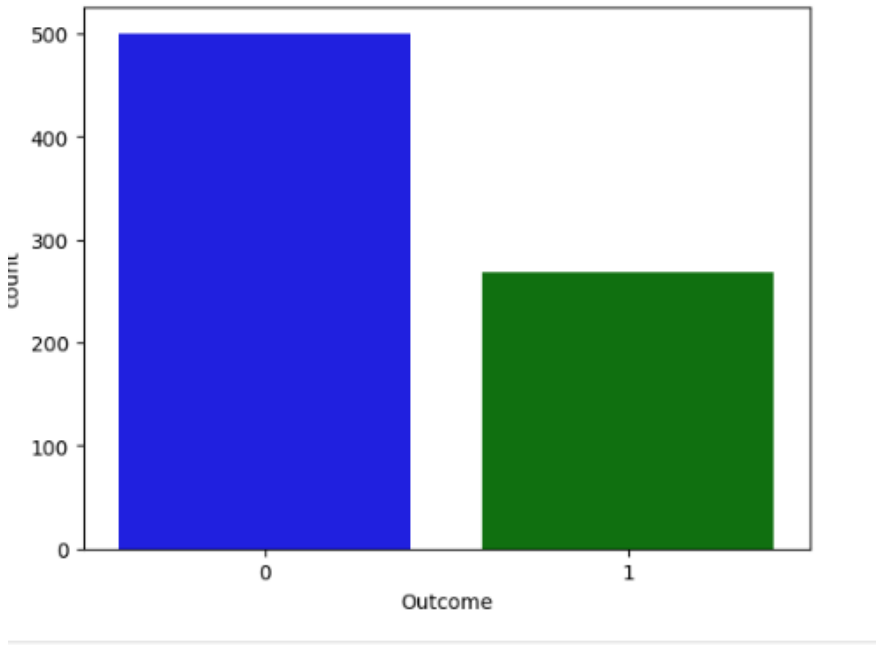


Рисунок 1.5 – Распределение по классам

Далее набор данных необходимо разделить на тестовую и тренировочную выборки и обучить модель, вывести матрицу запутанности (рисунок 1.6).

```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .2)
model = SVC()
model.fit(X_train , y_train)
predict_model = model.predict(X_test)
accuracy = accuracy_score(y_test, predict_model.round())
recall = recall_score(predict_model.round(), y_test)
f1 = f1_score(predict_model.round(), y_test)
sns.heatmap(confusion_matrix(predict_model.round(), y_test), annot = True)
print(model)
print(f"accuracy is : {accuracy} \nrecall is : {recall} \nf1 is : {f1}")
```

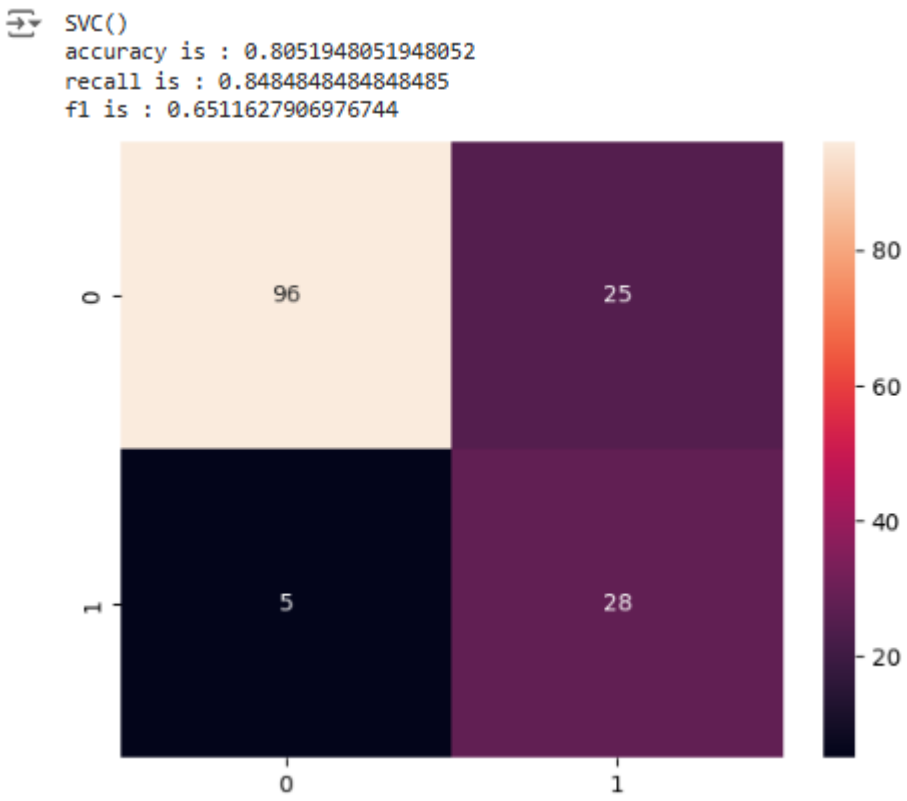


Рисунок 1.6 – Матрица запутанности

Так как набор не сбалансирован, необходимо использовать методы балансировки (RandomOverSampler и RandomUnderSampler), после чего обучить модель, вывести метрики и матрицу запутанности (рисунок 1.7).

Параметры RandomOverSampler:

- sampling_strategy: управляет балансировкой классов путем определения соотношения между количеством образцов в каждом классе;
- random_state: параметр для инициализации генератора случайных чисел для воспроизводимости результатов

Параметры RandomUnderSampler :

- random_state: параметр для инициализации генератора случайных чисел для воспроизводимости результатов. По умолчанию равен «None»;
- replacement: Опциональный параметр, указывающий, следует ли использовать замещение при сэмплировании;
- sampling_strategy: Пользовательский параметр для установки желаемого соотношения классов в итоговом наборе данных. По умолчанию

равен «auto», что означает, что соотношение будет автоматически настроено на равное количество примеров в каждом классе.

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=0)
X_resampled, y_resampled = ros.fit_resample(X, y)
from collections import Counter
print(sorted(Counter(y_resampled).items()))
```

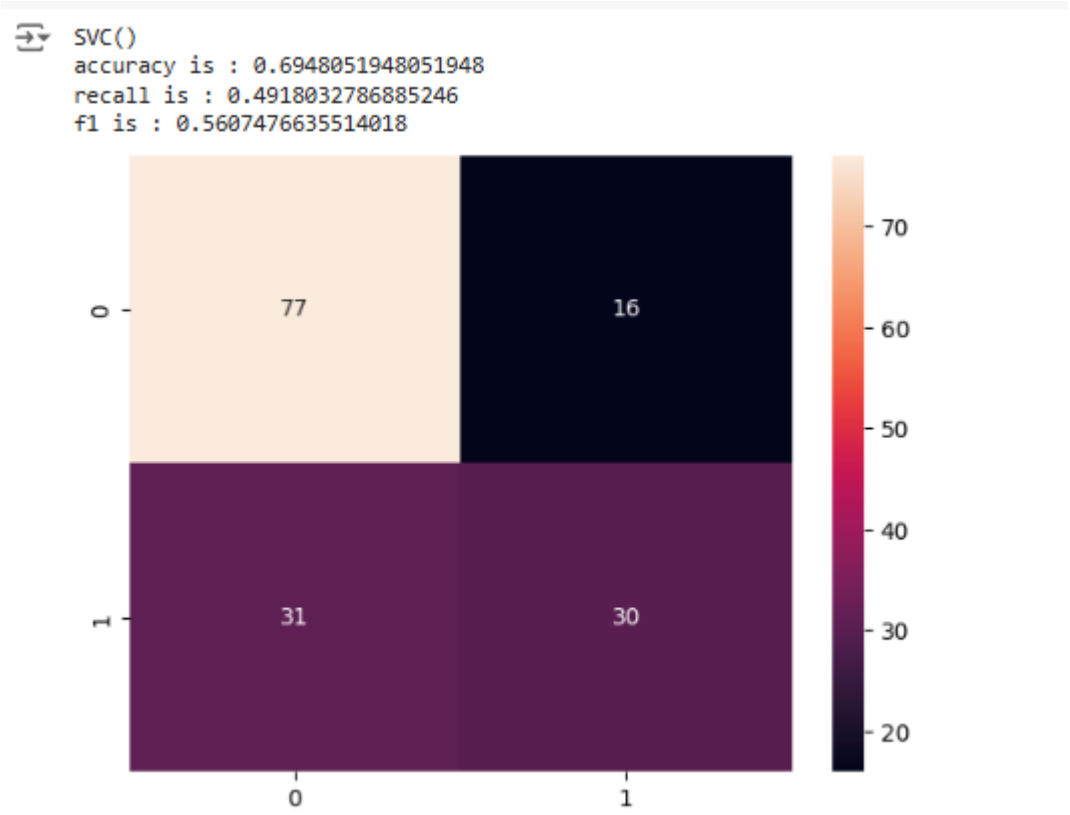


Рисунок 1.7 - Матрица запутанности

2 Задание на лабораторную работу

1. Ознакомиться с методическими указаниями, а также представленными примерами в блокноте Google Colab.

2. Провести анализ данных, при необходимости используя опыт предыдущий лабораторных провести кодирование переменных и обучить модель.

3. Провести балансировку данных и обучить модель на сбалансированных данных. Свести в таблицу результаты до/после и сравнить их. При балансировке необходимо подобрать параметры, при которых модель покажет лучшие показатели.

4. Написать отчет в соответствии с ОС ТУСУР.

В таблице 2.1 представлены варианты индивидуального задания

Таблица 2.1 – Индивидуальное задание

Вар.	Набор данных
1	Diabetes prediction dataset (kaggle.com)
2	Diabetes Dataset (kaggle.com)

Контрольные вопросы

1. Какие существуют способы борьбы с несбалансированными данными?
2. Что такое oversampling?
3. Что такое undersampling?
4. Что показывает матрица запутанности?
5. Каким образом определить распределение по классам?

ЛАБОРАТОРНАЯ РАБОТА №6

Разработка модели ИИ с использованием Weka

Целью данной работы является исследование базовых методов работы с программным обеспечением WEKA.

Краткие теоретические сведения

WEKA - программное обеспечение с открытым исходным кодом, предоставляющее инструменты для предварительной обработки данных, реализации нескольких алгоритмов машинного обучения и инструменты визуализации, позволяющие разрабатывать методы машинного обучения и применять их к реальным задачам интеллектуального анализа данных.

На рисунке 1 представлена структура WEKA.

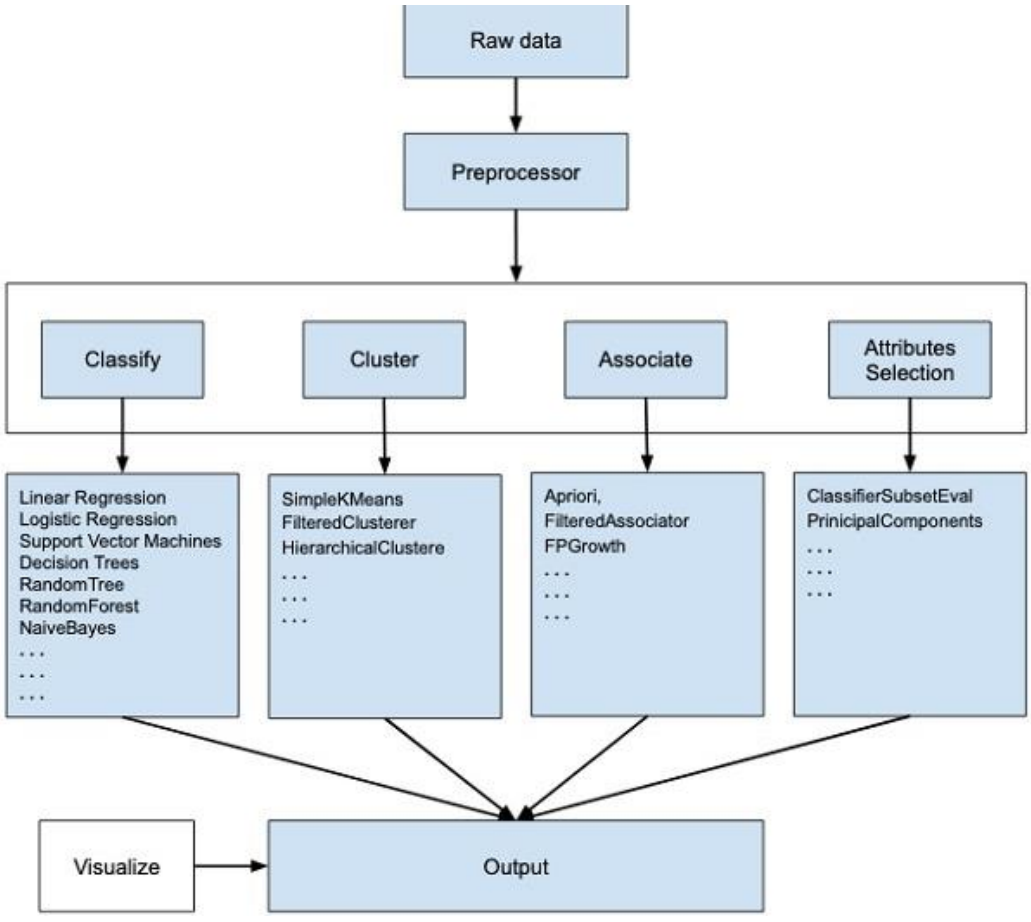


Рисунок 1 – Схема WEKA

Weka предоставляет набор стандартных методов обработки данных и вывода, таких как:

Предварительная обработка данных: после загрузки набора данных Weka позволяет быстро изучить его атрибуты и экземпляры. Кроме того, доступны различные методы фильтрации, например, для преобразования категориальных данных в числовые и т.д.

Классификация и регрессионные алгоритмы: набор различных алгоритмов, таких как наивный байесовский алгоритм, деревья решений,метод K-ближайших соседей, методы ансамблей и различные варианты линейной регрессии.

Кластеризация: этот метод может быть использован для того, чтобы определить основные категории в данных.

Обнаружение ассоциаций: обнаружение правил в наборе данных, чтобы упростить выявление закономерностей и связей между различными признаками.

Выбор признаков: позволяет уменьшить размерность набора данных (например, для ускорения времени обучения и производительности).

Визуализация данных: набор интегрированных методов для быстрой визуализации корреляций между функциями и представления изученных шаблонов машинного обучения, таких как деревья решений и кластеризация K-средних.

1 Ход работы

1.1 Предварительная обработка данных

Для загрузки данных необходимо нажать кнопку импорта данных (рисунок 1.1) и выбрать файл (рисунок 1.2).

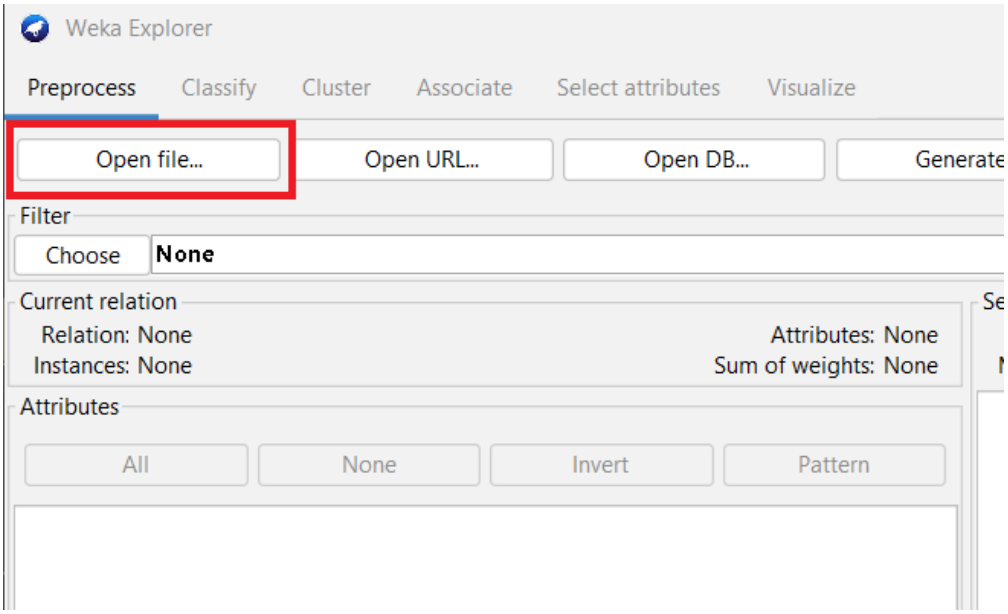


Рисунок 1.1 – Выбор источника загрузки

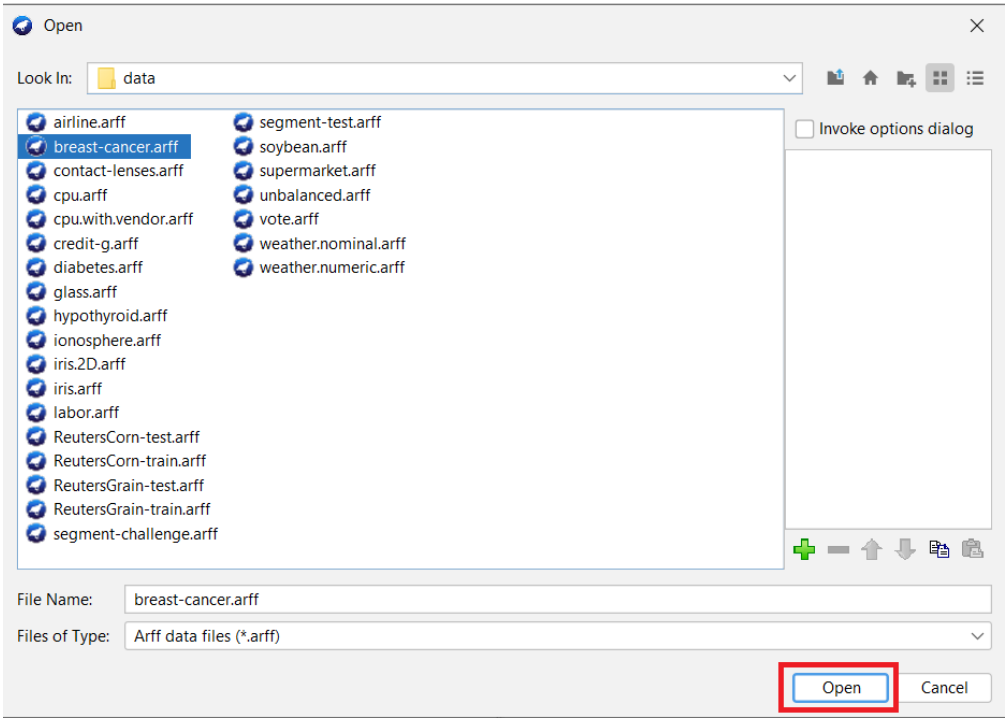


Рисунок 1.2 – Выбор файла для загрузки

После загрузки появится информация о наборе. На рисунке 1.3 представлено количество объектов, атрибутов и название набора.

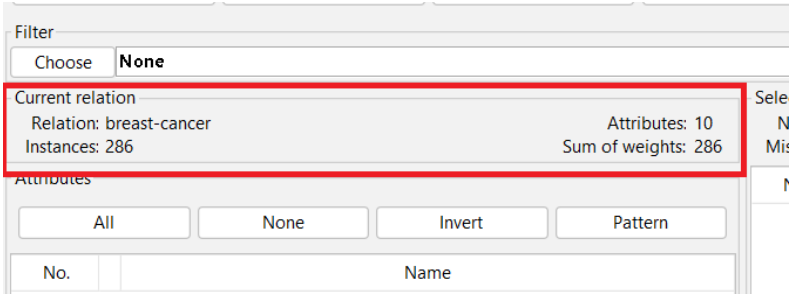


Рисунок 1.3 – Информация о наборе данных

Информацию об одном атрибуте можно посмотреть в специальном окне выбрав его из списка (рисунок 1.4).

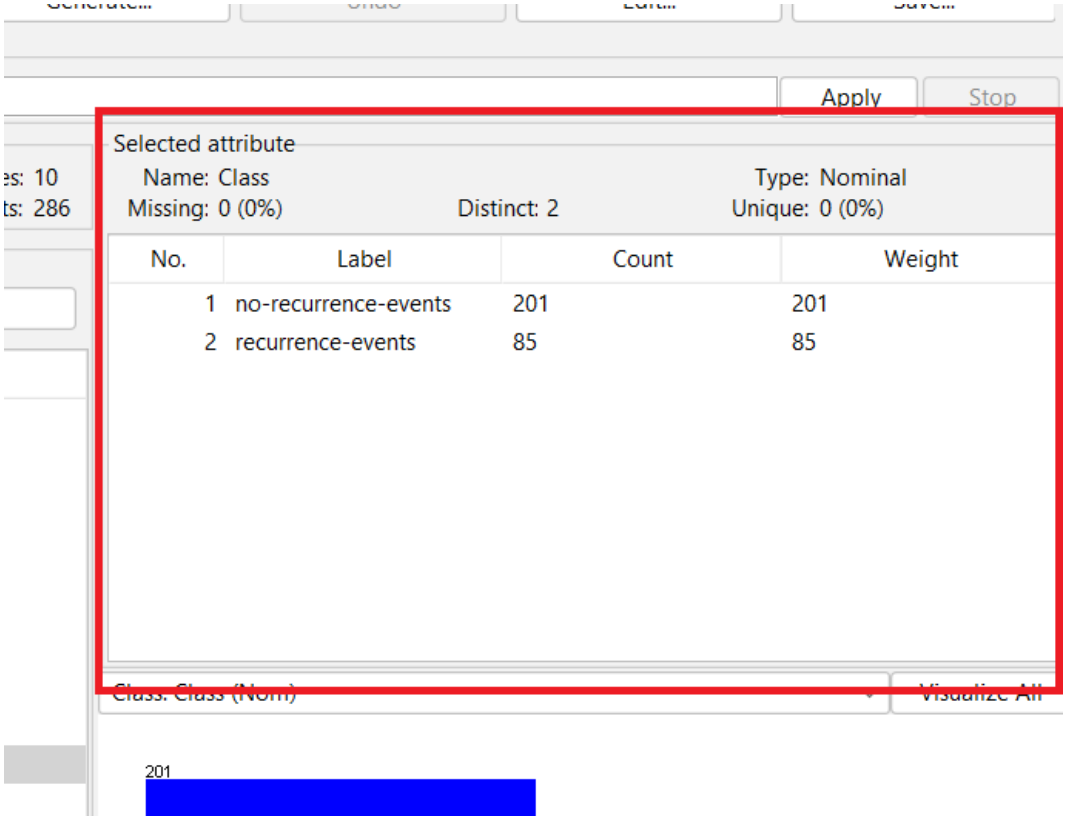


Рисунок 1.4 – Информация об атрибуте

В правом нижнем углу находится окно визуализации, в котором при выборе атрибута отражается график его распределения (рисунок 1.5).

Рисунок 1.5 – График для одного атрибута

1.2 Классификация

Для того чтобы выполнить классификацию данных необходимо выбрать вариант тестирования (рисунок 1.6), выходной класс (рисунок 1.7), алгоритм классификации (рисунок 1.8) и нажать старт.

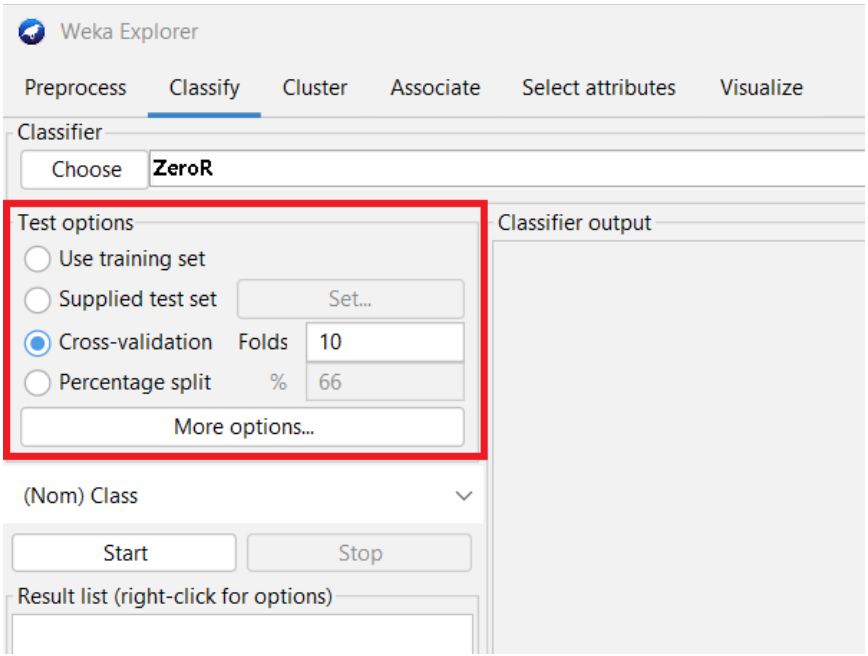


Рисунок 1.6 – Вариант тестирования

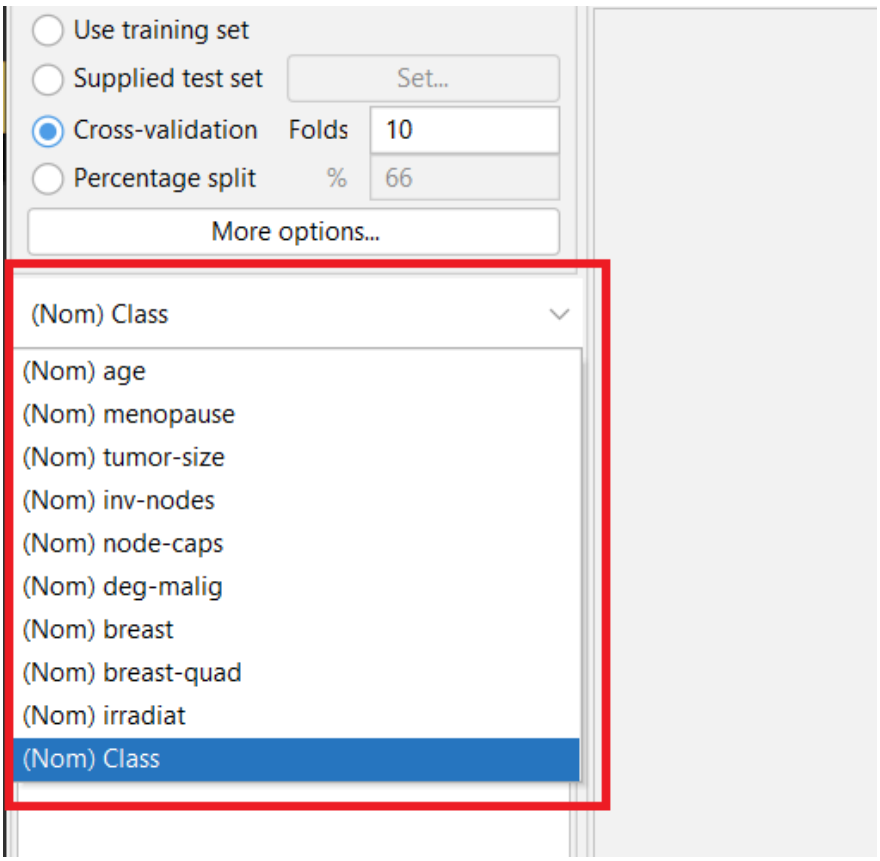


Рисунок 1.7 – Выбор выходного класса

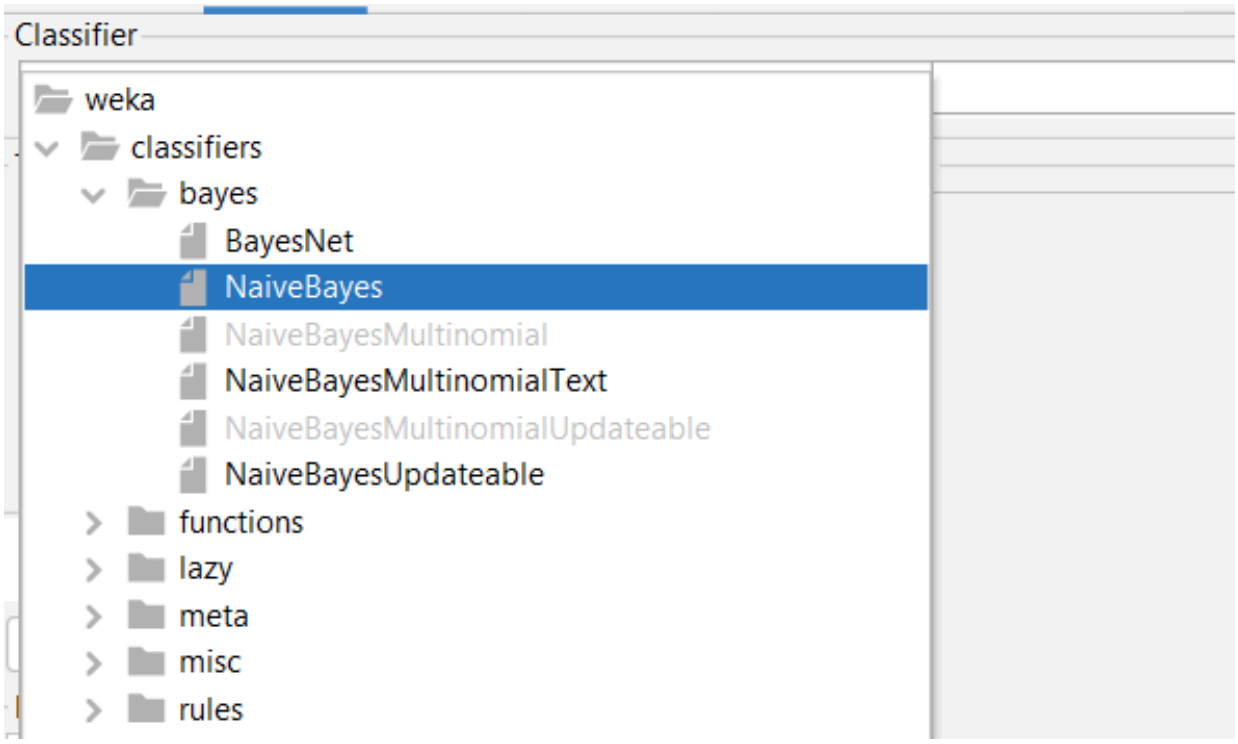


Рисунок 1.8 – Выбор классификатора

Результаты классификации будут представлены в специальном окне, в котором отражены метрики для оценки качества модели и матрица запутанности (рисунок 1.9).

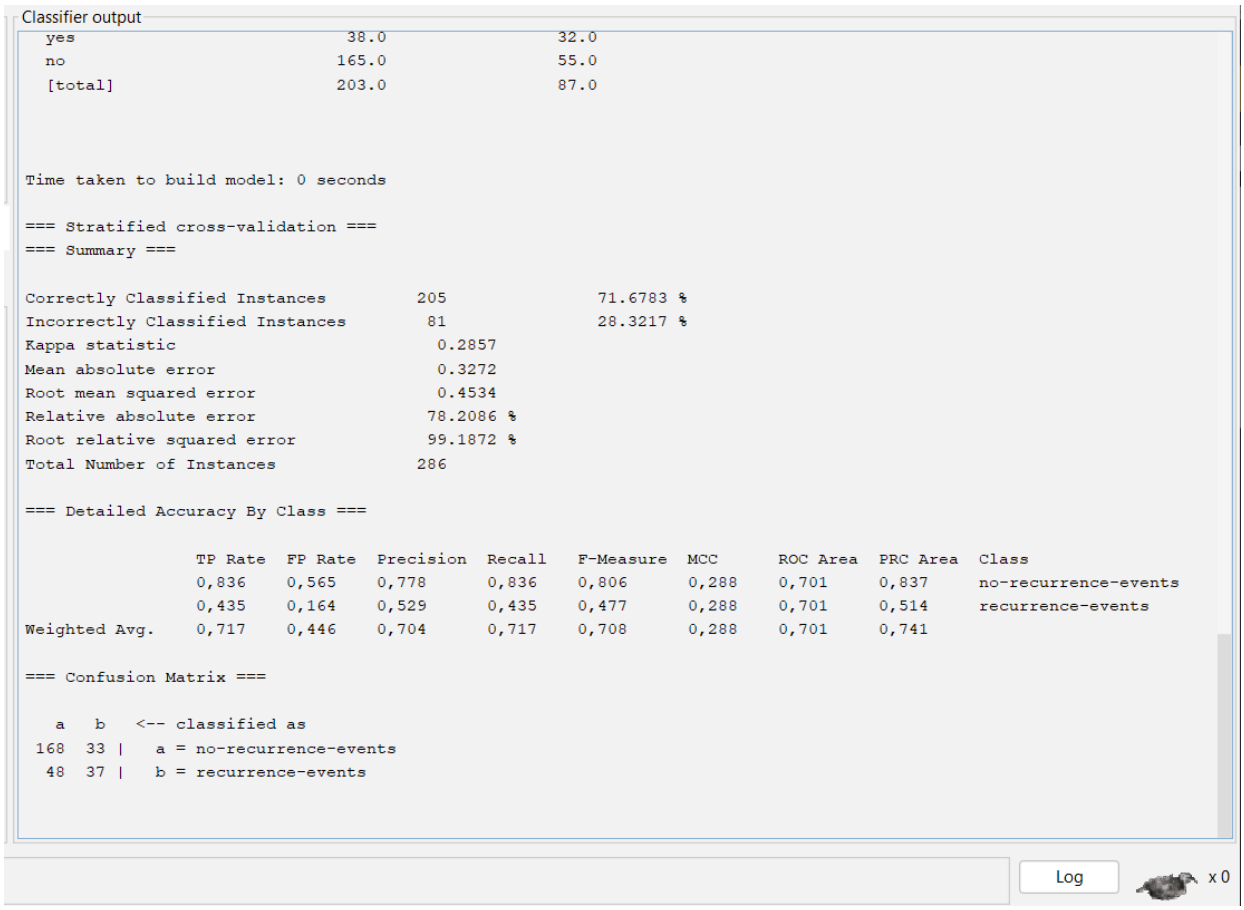


Рисунок 1.9 – Результаты классификации

Для того чтобы получить график ошибок классификатора необходимо вызвать контекстное меню и выбрать «visualize classiffier errors». На рисунках 1.10 и 1.11 представлен порядок получения графика.

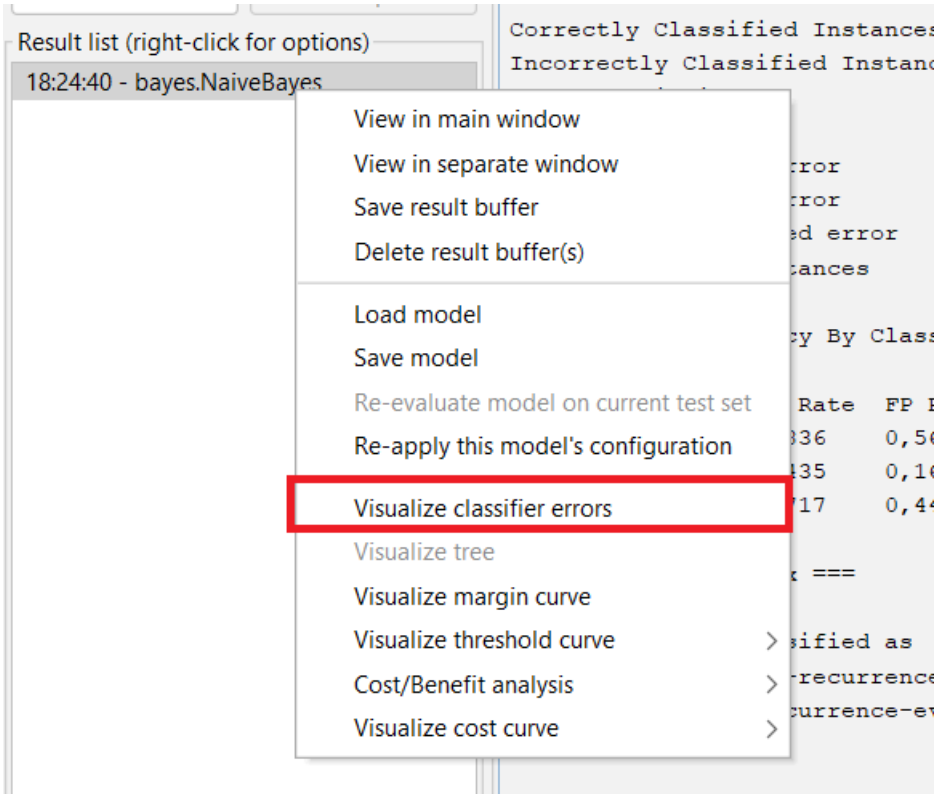


Рисунок 1.10 – Контекстное меню результата

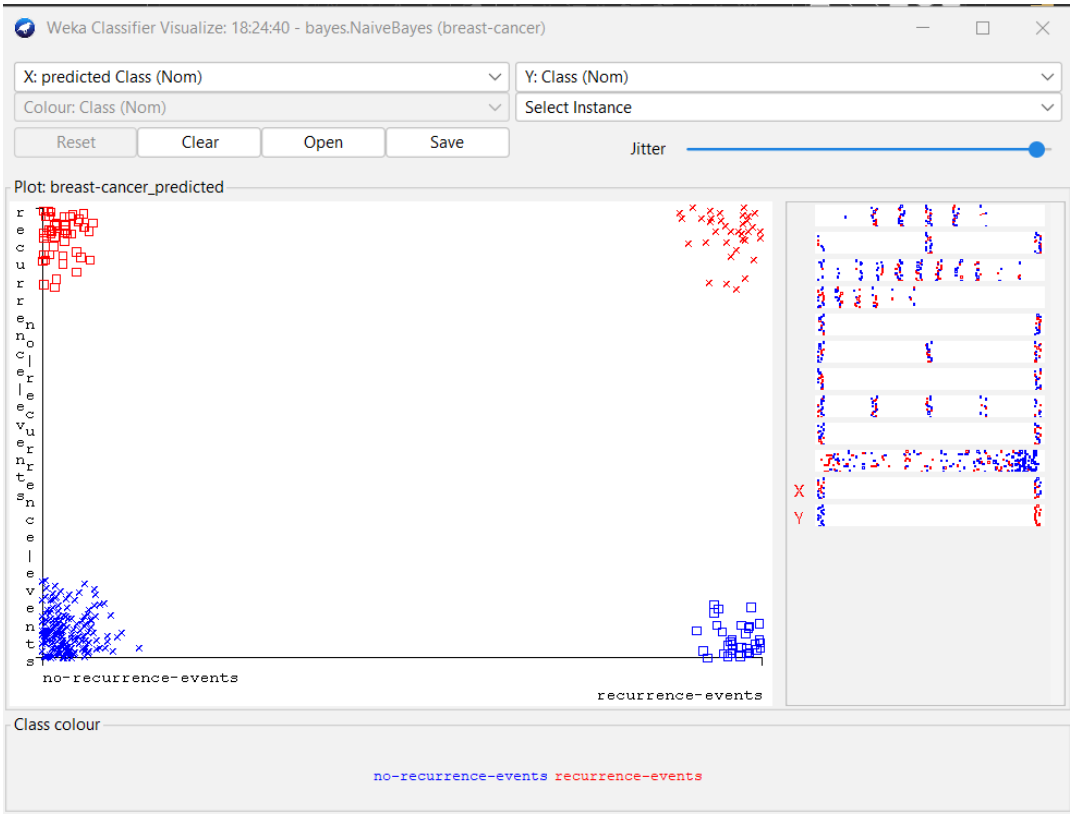


Рисунок 1.11 – График ошибок классификатора

1.3 Кластеризация

Кластеризация – это метод машинного обучения, который используется для разделения набора данных на группы или кластеры на основе их сходства. Для того чтобы выполнить кластеризацию необходимо выбрать режим кластера (рисунок 1.12), алгоритм кластеризации (рисунок 1.13) и нажать старт.

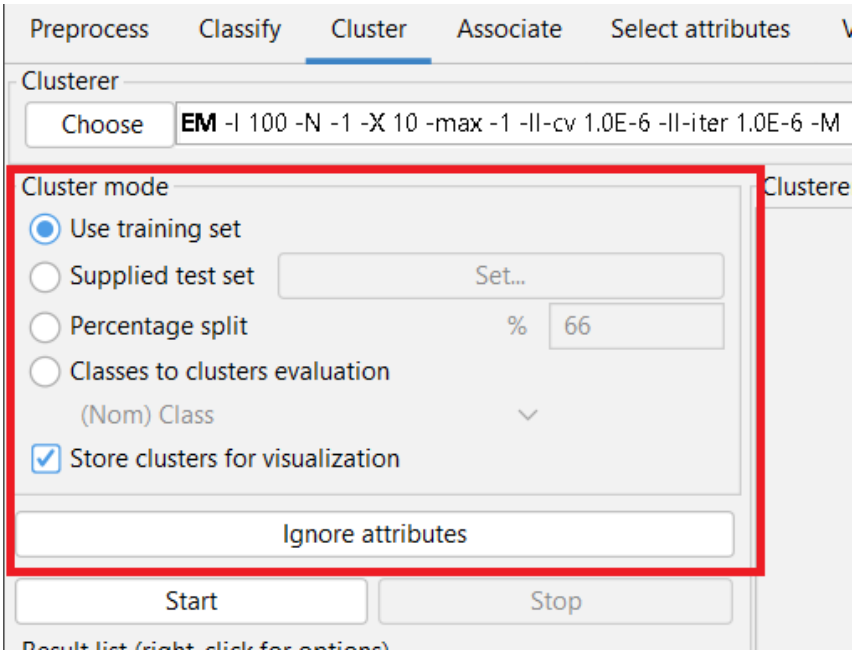


Рисунок 1.12 – Режим кластера

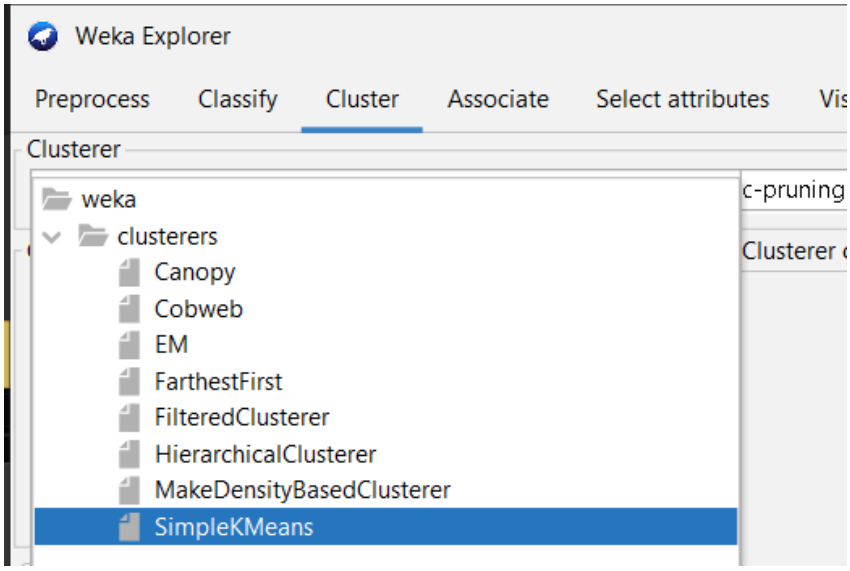


Рисунок 1.13 – Выбор алгоритма кластеризации

После выполнения алгоритма, WEKA предоставит результаты кластеризации (рисунок 1.14), включая распределение данных по кластерам, центры кластеров и визуализацию результатов.

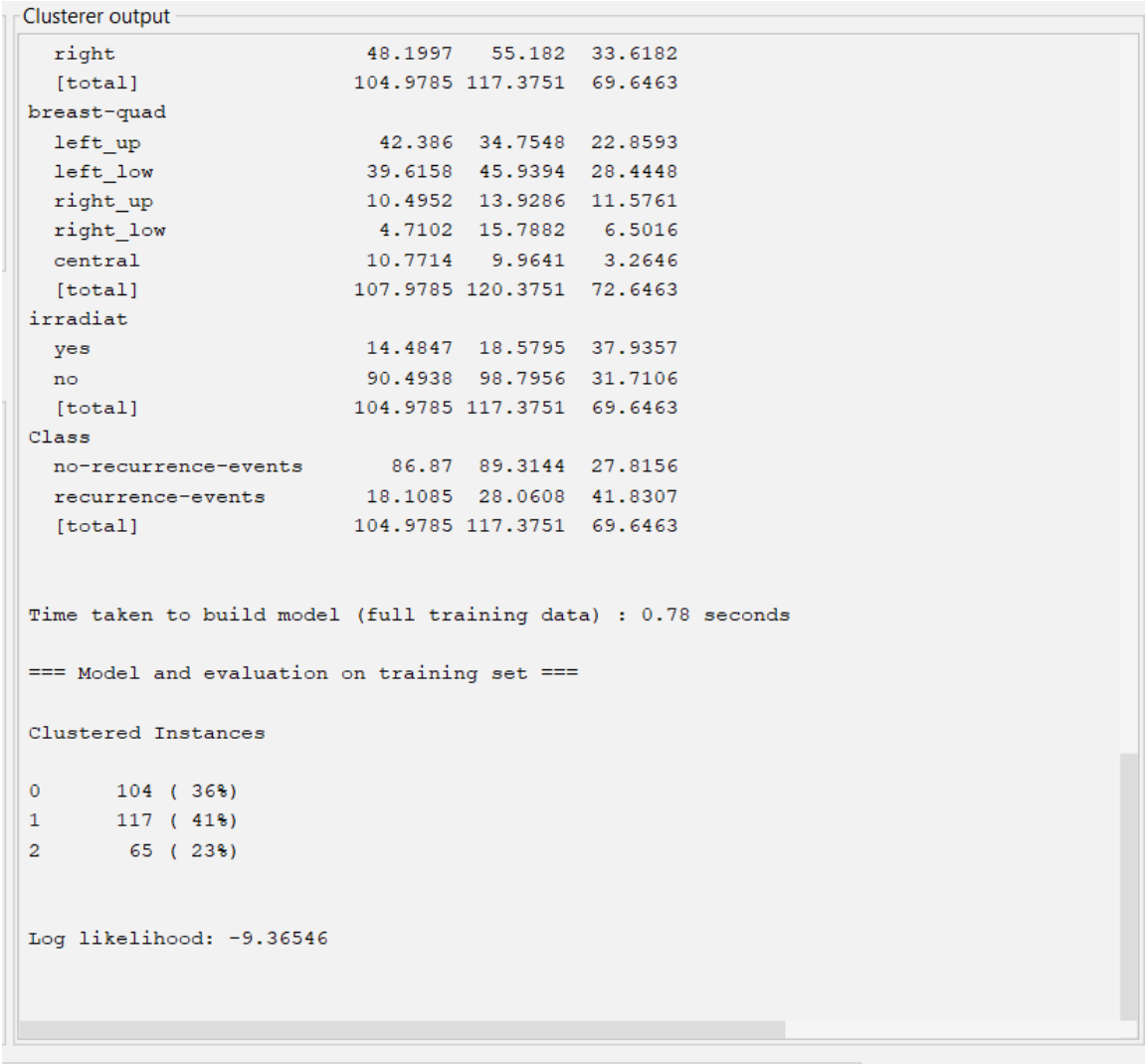


Рисунок 1.14 – Результаты кластеризации

Для того чтобы получить график распределения кластеров необходимо вызвать контекстное меню и выбрать «visualize claster assignments». На рисунках 1.14 и 1.15 представлен порядок получения графика.

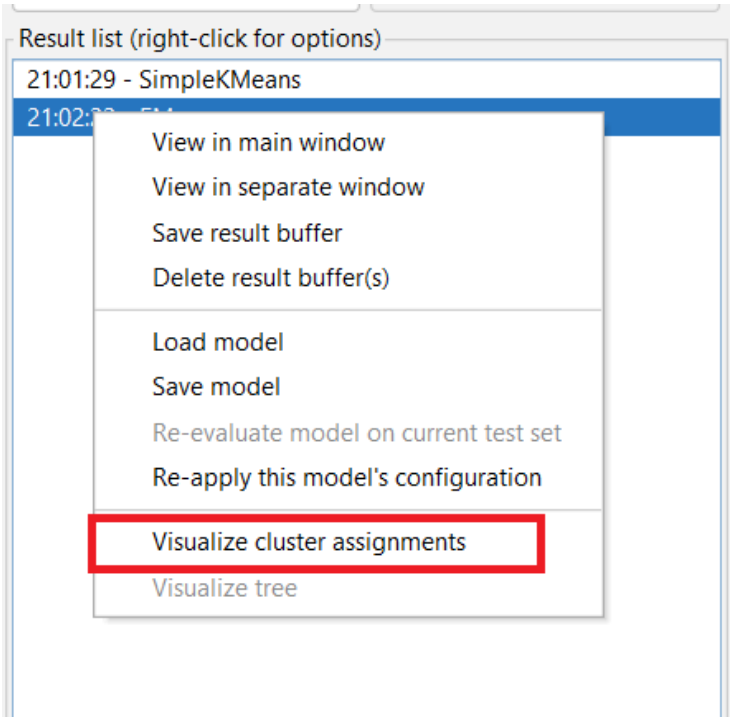


Рисунок 1.15 – Контекстное меню результата

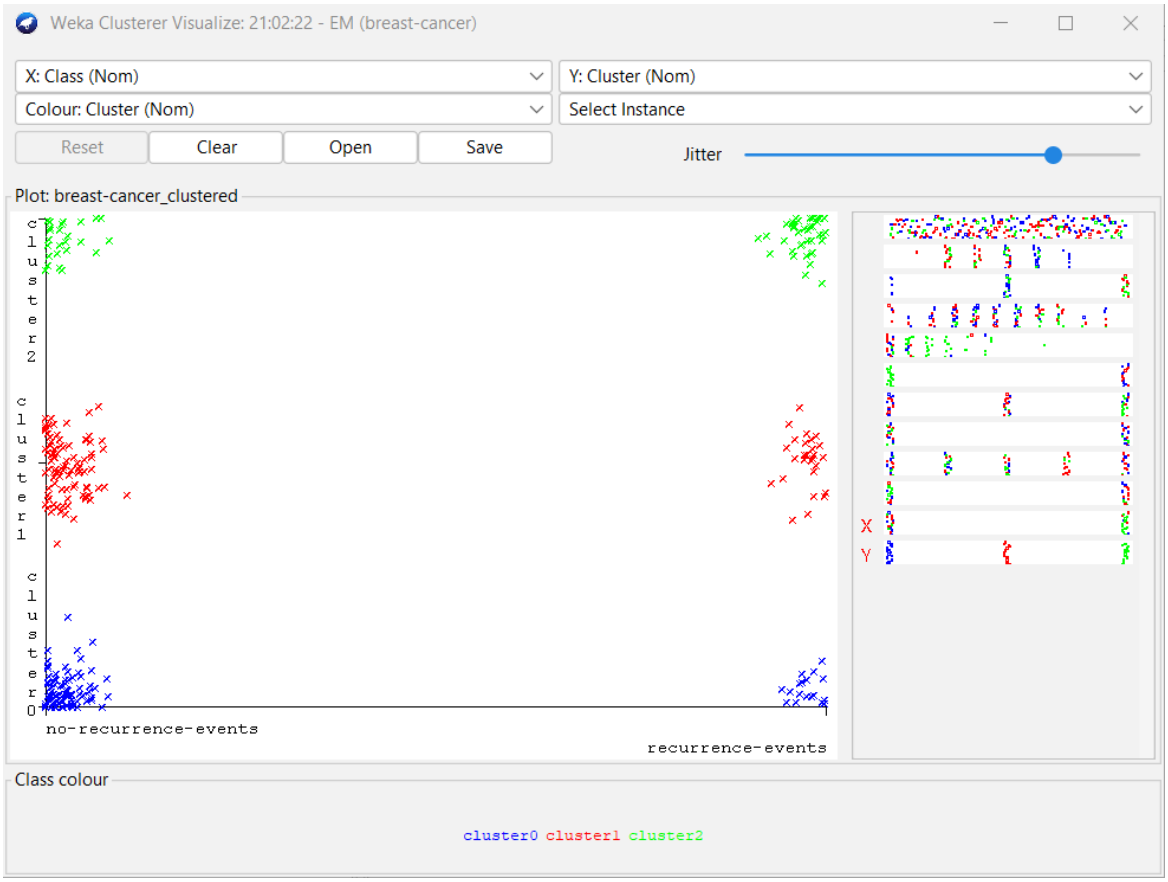


Рисунок 1.16 – Визуализация результатов

1.4 Ассоциация

Для того чтобы выполнить ассоциацию необходимо выбрать алгоритм (рисунок 1.17), изучить его параметры (рисунок 1.18) и нажать старт.

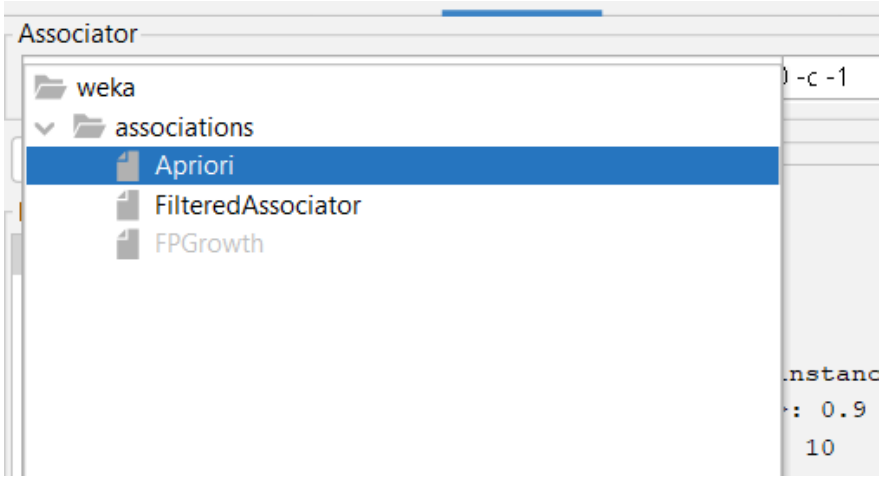


Рисунок 1.17 – Выбор алгоритма

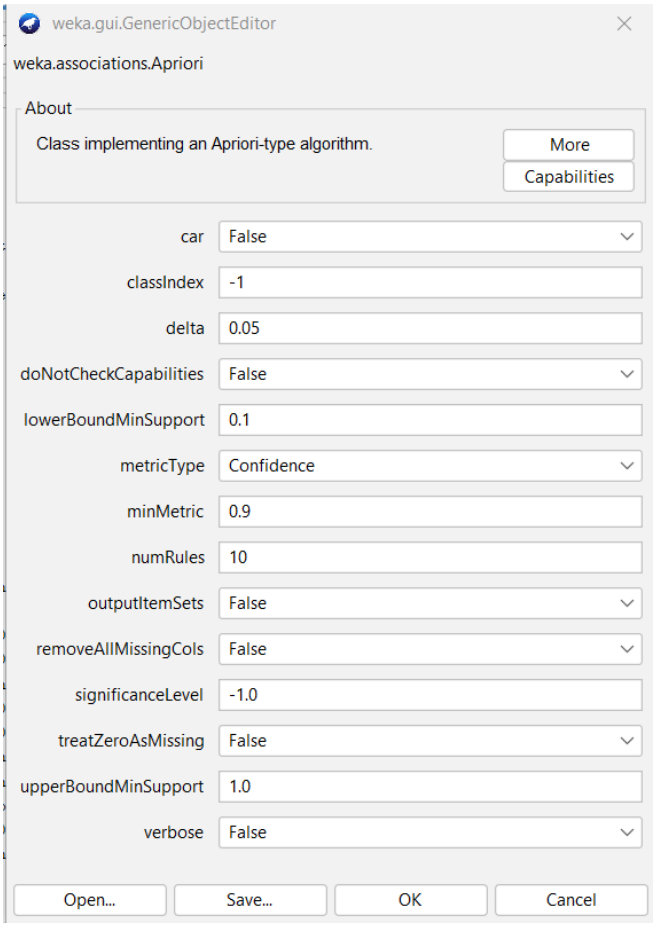


Рисунок 1.18 – Параметры алгоритма

После выполнения будут представлены результаты выполнения (рисунок 1.19) включая лучшие правила, минимальную поддержку и минимальную достоверность, а также количество экземпляров с минимальной поддержкой и количество циклов.

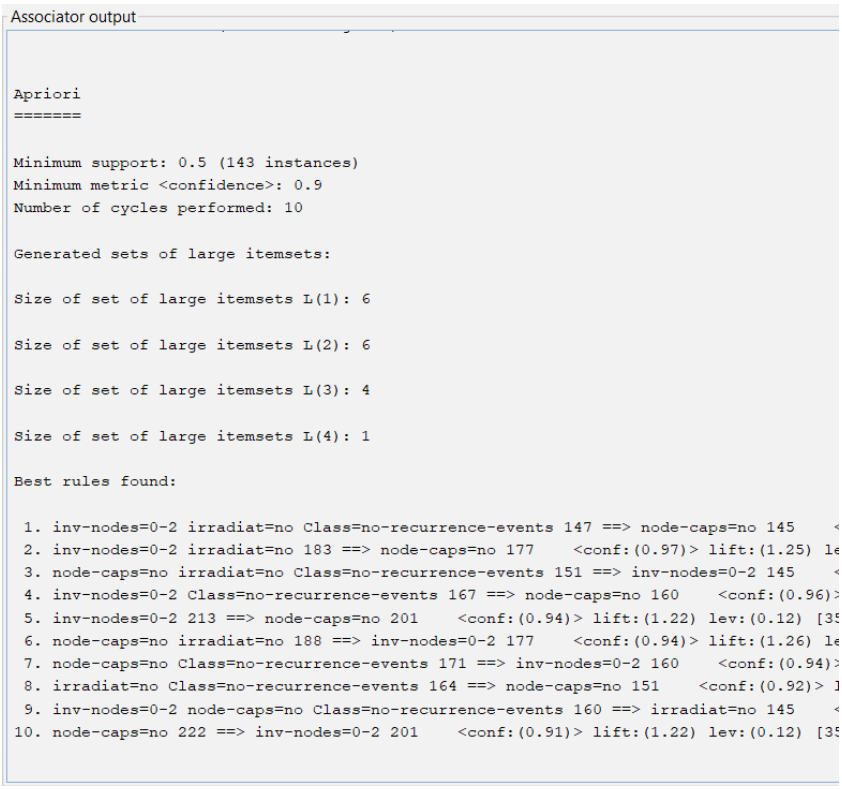


Рисунок 1.19 – Результаты выполнения алгоритма

1.5 Выбор атрибутов

Для того чтобы уменьшить набор и оставить только важные признаки необходимо выбрать вычислитель атрибутов (рисунок 1.20), метод поиска (рисунок 1.21), выходной класс и нажать старт.

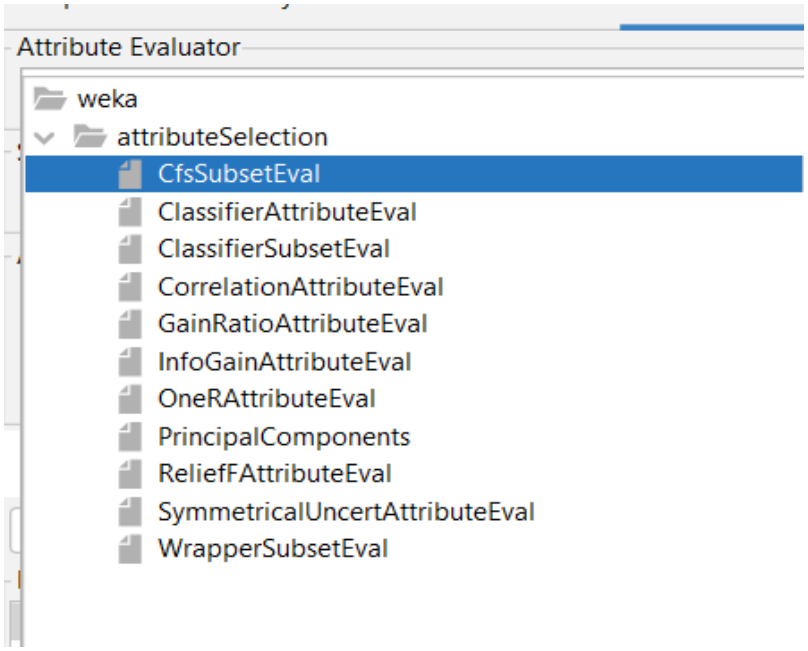


Рисунок 1.20 – Выбор вычислителя

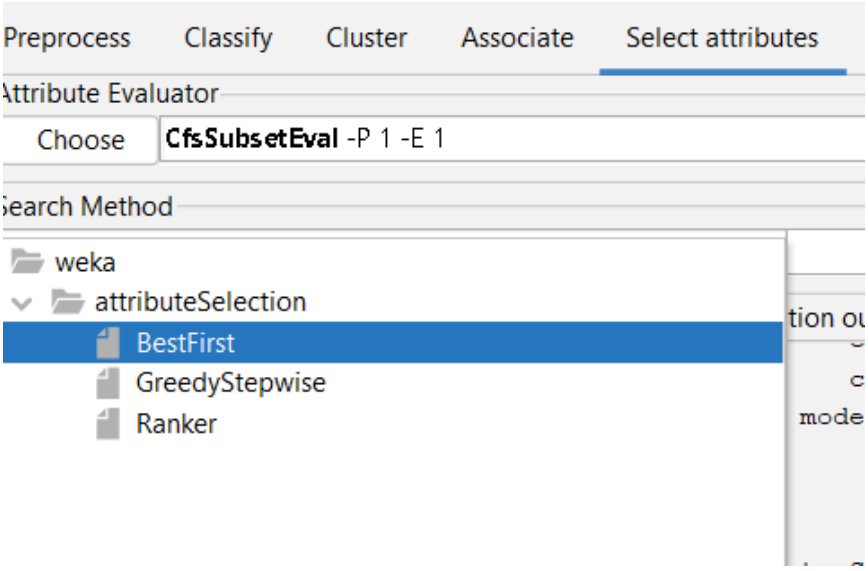


Рисунок 1.21 – Выбор метода поиска

После этого будут выведены результаты (рисунок 1.22), которые включают метод поиска, начальный набор, направление поиска, общее количество оцененных подмножеств, оценку наилучшего найденного подмножества и выбранные атрибуты.

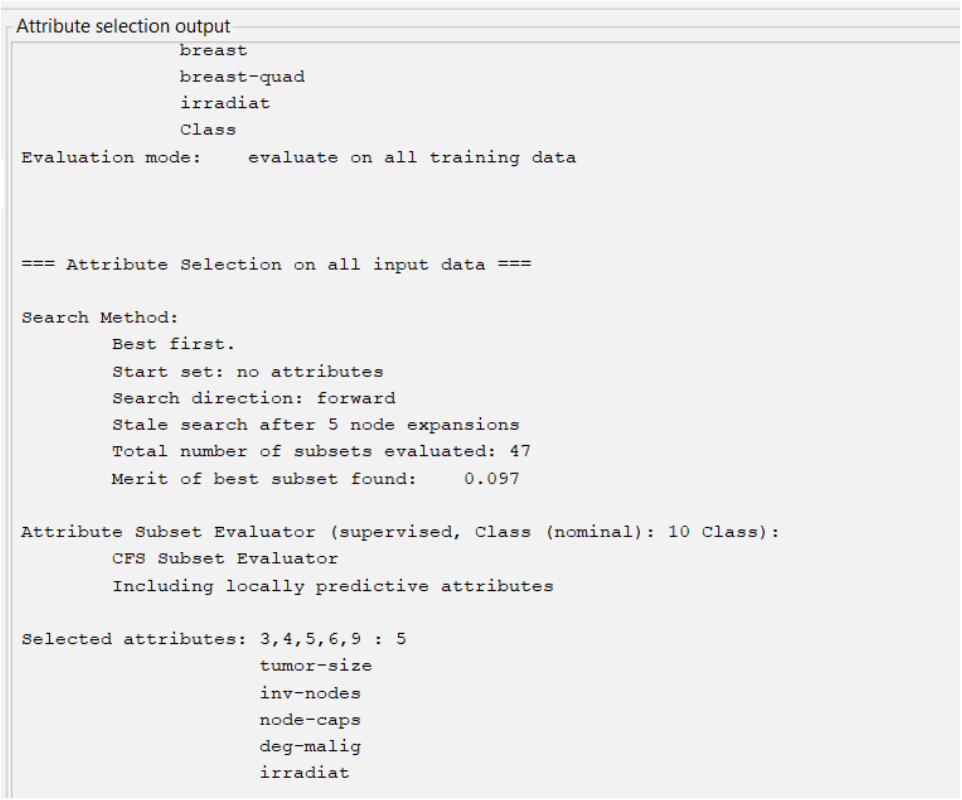


Рисунок 1.22 – Результаты выбора атрибутов

2 Задание на лабораторную работу.

1. Загрузить набор данных согласно варианту.
2. Выполнить ход работы.
3. Проанализировать структуру данных, включая количество объектов, атрибутов, классов, а также статистику и график для одного из атрибутов.
4. Выполнить классификацию данных и оценить качество модели на тестовой выборке, используя метрики, такие как точность, полнота, F-мера, а также отразить график ошибок классификатора.
5. Провести кластеризацию данных, сделать выводы на основе полученных результатов, а также привести графики.
6. Проанализировать результаты выполнения алгоритма ассоциации включая минимальную поддержку и минимальную достоверность, а также

количество экземпляров с минимальной поддержкой и количество циклов. Привести лучшие правила и их толкование.

7. Изучить результаты отбора признаков и отразить в выводе: метод поиска, начальный набор, направление поиска, общее количество оцененных подмножеств, оценку наилучшего найденного подмножества и выбранные атрибуты

8. Написать отчет в соответствии с ОС ТУСУР.

В таблице 2.1 представлены варианты индивидуального задания

Таблица 2.1 – Индивидуальное задание

Вар.	**Классификация**	**Кластеризация**	**Выбор признаков**	**Набор данных**
1	LinerRegression, ZeroR	EM	ClassifierAttributeEval, Ranker	Diabetes Dataset (kaggle.com)
2	SMOreg, RandomForest	SimpleKMeans	CfsSubsetEval, BestFirst	Heart Disease Dataset (kaggle.com)
3	RandomForest, RandomTree	HierarchicalClusterer	InfoGainAttributeEval, Ranker	Diabetes Dataset (kaggle.com)
4	LinerRegression, LWL	FarthestFirst	WrapperSubsetEval, GreedyStepwise	Heart Disease Dataset (kaggle.com)
5	Gaussian Processes, RandomTree	Canopy	ReliefFAttributeEval, Ranker	Diabetes Dataset (kaggle.com)

Контрольные вопросы

1. Что такое WEKA? Какие инструменты по работе с данными включает в себя?

2. Каким образом можно загрузить и предобработать данные в Weka?

3. Какие типы задач можно решать с помощью Weka?

4. Какие возможности есть для визуализации данных и результатов работы с WEKA?

5. Какие метрики используются для оценки качества модели в WEKA?

ЛАБОРАТОРНАЯ РАБОТА №7

Использование rapid miner для работы с медицинскими данными пациента

Целью данной лабораторной работы является изучение возможностей использования программного обеспечения RapidMiner для анализа медицинских данных пациента.

Краткие теоретические сведения

RapidMiner — это программная многопользовательская платформа, которая представляет собой интегрированную среду для обработки данных в больших информационных массивах, машинного обучения, текстовой аналитики и построения прогностических моделей, а также для решения иных задач работы с данными.

RapidMiner поддерживает различные типы данных, такие как числовые, категориальные, временные ряды и текстовые данные. Он также предоставляет большое количество встроенных алгоритмов машинного обучения, включая классификацию, регрессию, кластеризацию и другие.

1 Ход работы

1.2 Загрузка и анализ данных

Для загрузки данных необходимо нажать кнопку импорта данных (рисунок 1.1), выбрать источник загрузки (рисунок 1.2) и файл (рисунок 1.3).

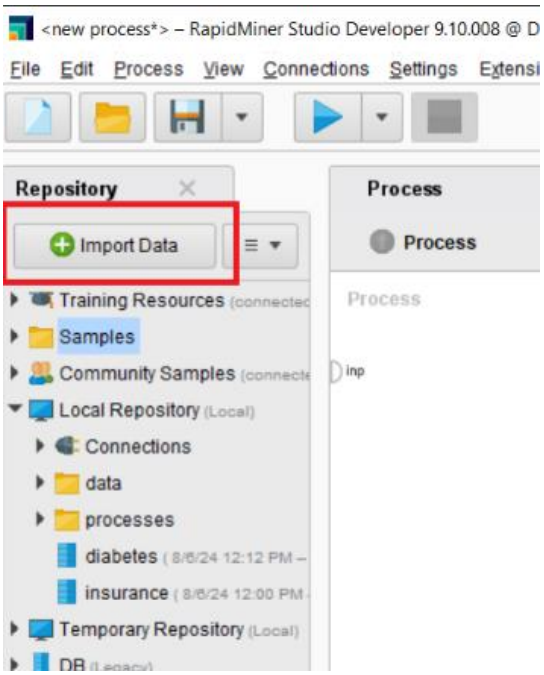


Рисунок 1.1 – Кнопка загрузки

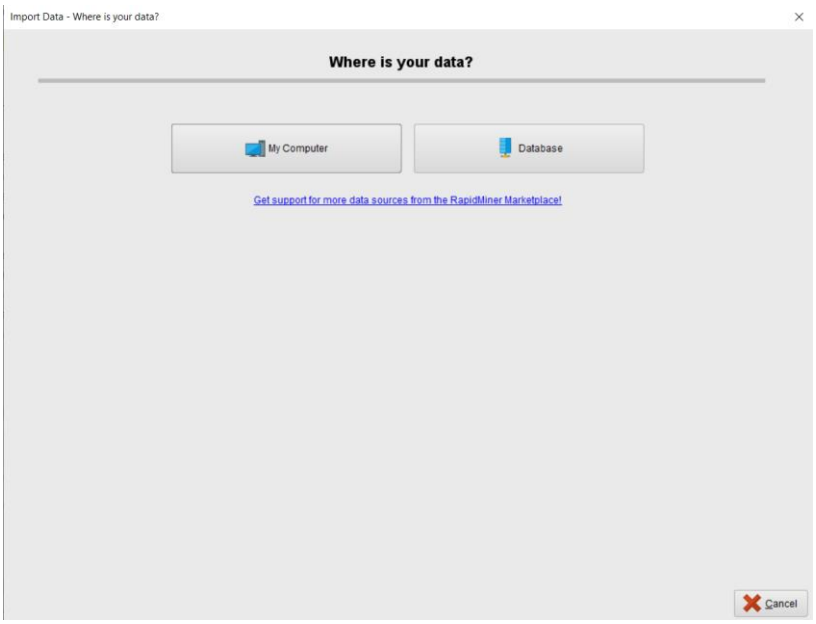


Рисунок 1.2 – Выбор источника загрузки

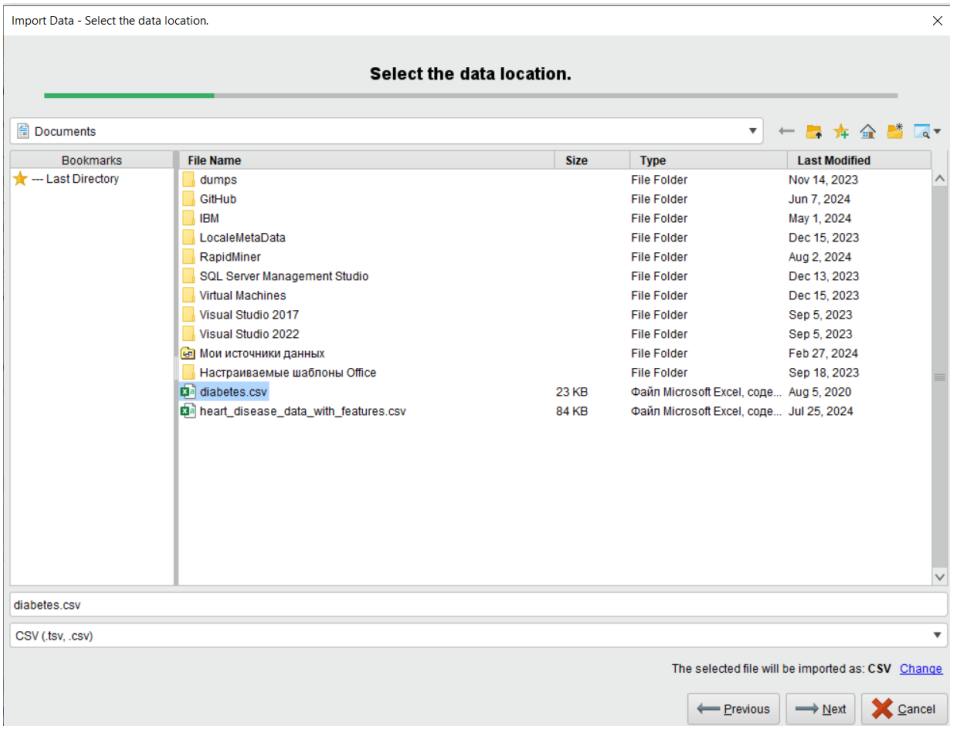


Рисунок 1.3 – Выбор файла для загрузки

После выбора файла необходимо указать в какой строке находятся заголовки и что используется в качестве разделителя колонок (рисунок 1.4), правильность формата для каждой колонки (рисунок 1.5), а также выбрать хранилище для сохранения результатов (рисунок 1.6).

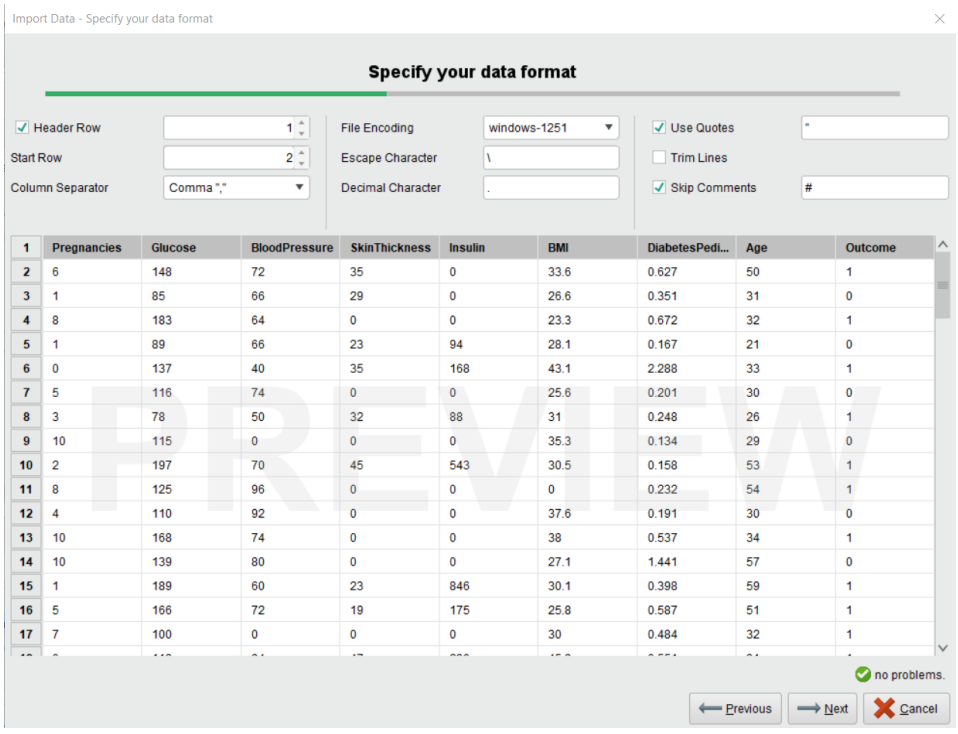


Рисунок 1.4 – Настройка загрузки файла

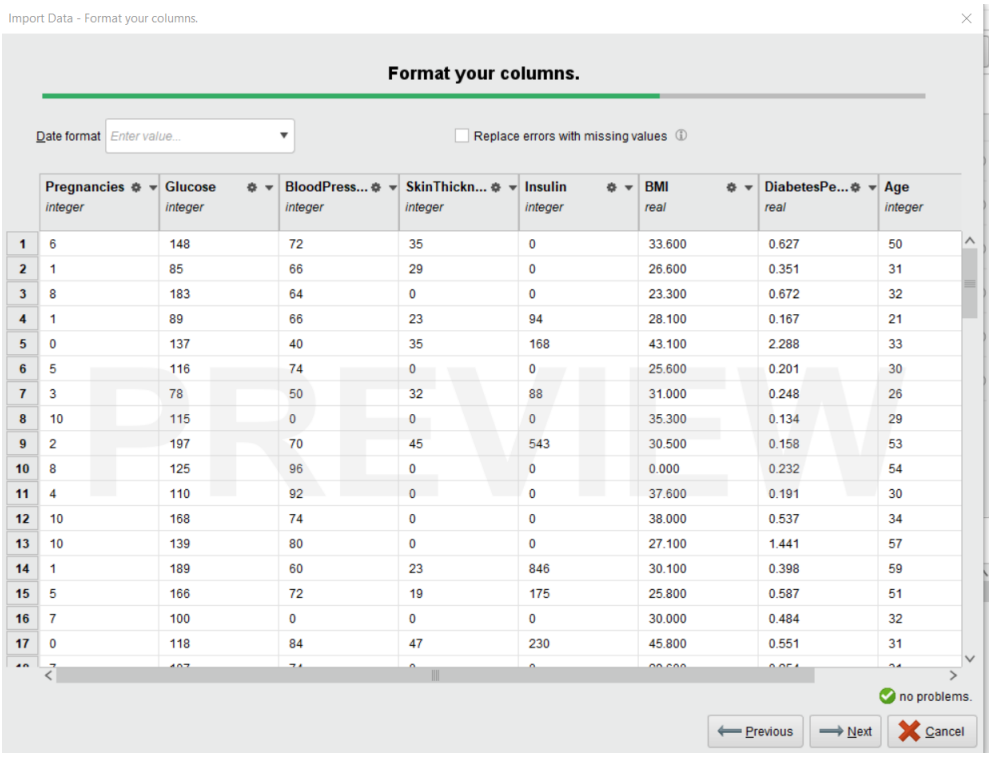


Рисунок 1.5 – Формат колонок

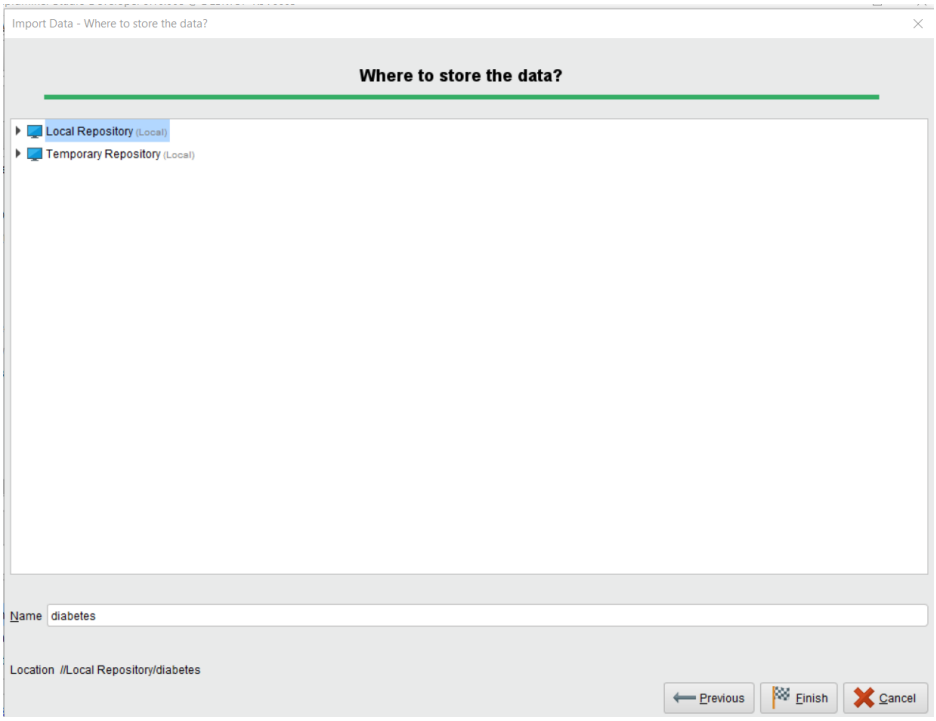


Рисунок 1.6 – Выбор хранилища

После загрузки данных открывается окно Results, в котором можно проанализировать структуру данных с помощью разделов статистика и визуализация (рисунок 1.7 – 1.8).

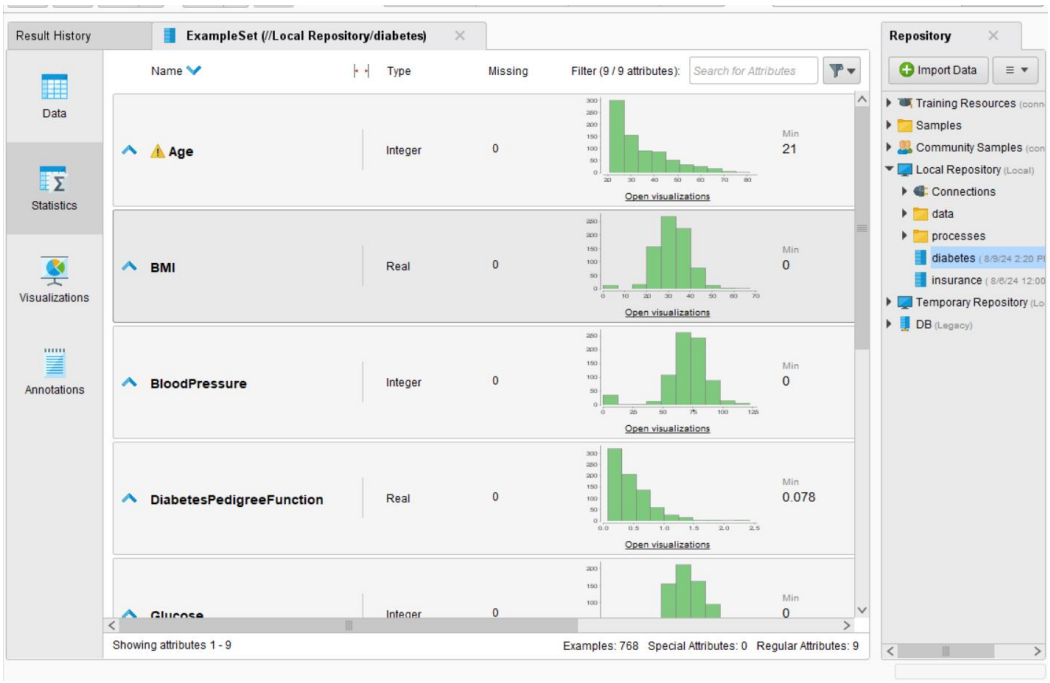


Рисунок 1.7 – Информация о наборе данных

Рисунок 1.8 – Пример визуализации

1.2 Auto Model

Auto Model — это расширение для RapidMiner, которое ускоряет процесс создания и проверки моделей.

Auto Model помогает оценить данные, предоставляет актуальные модели для решения задачи, и помогает сравнить результаты для этих моделей после завершения расчетов.

Для использования Auto Model необходимо выбрать набор данных (рисунок 1.9), задачу (рисунок 1.10), на следующем шаге нажать Next (рисунок 1.11).

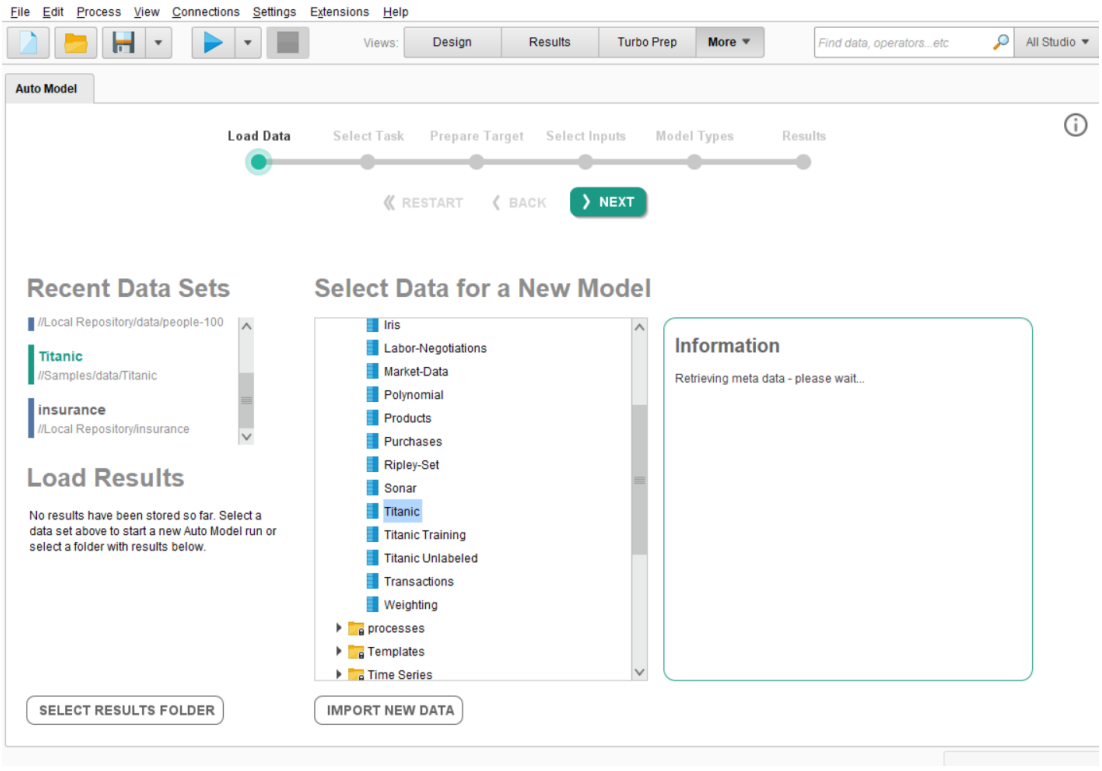


Рисунок 1.9 – Загрузка набора

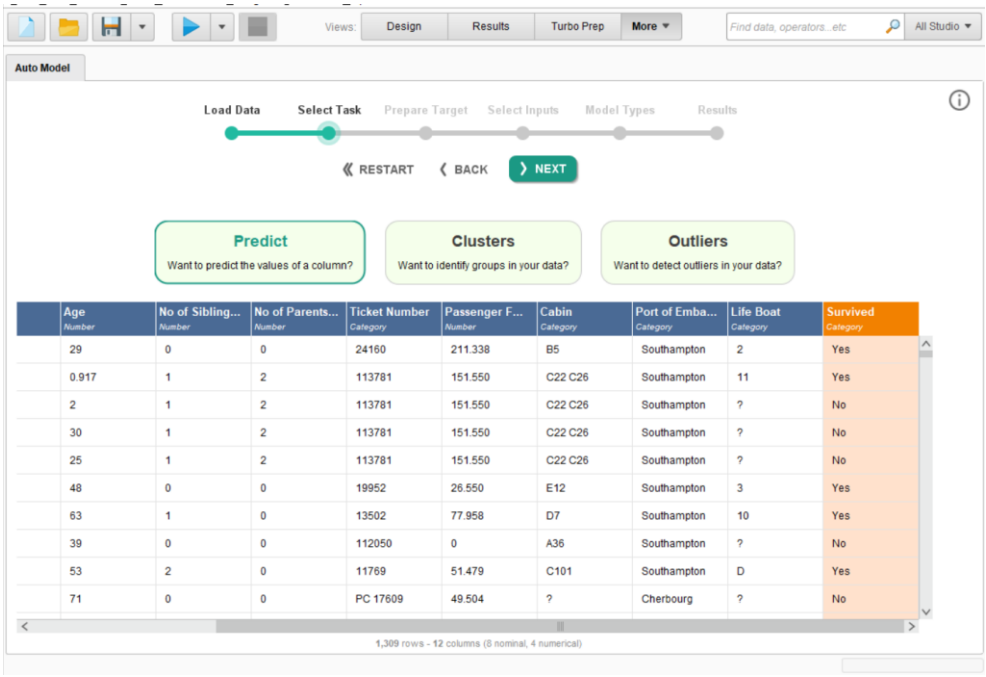


Рисунок 1.10 – Выбор задачи

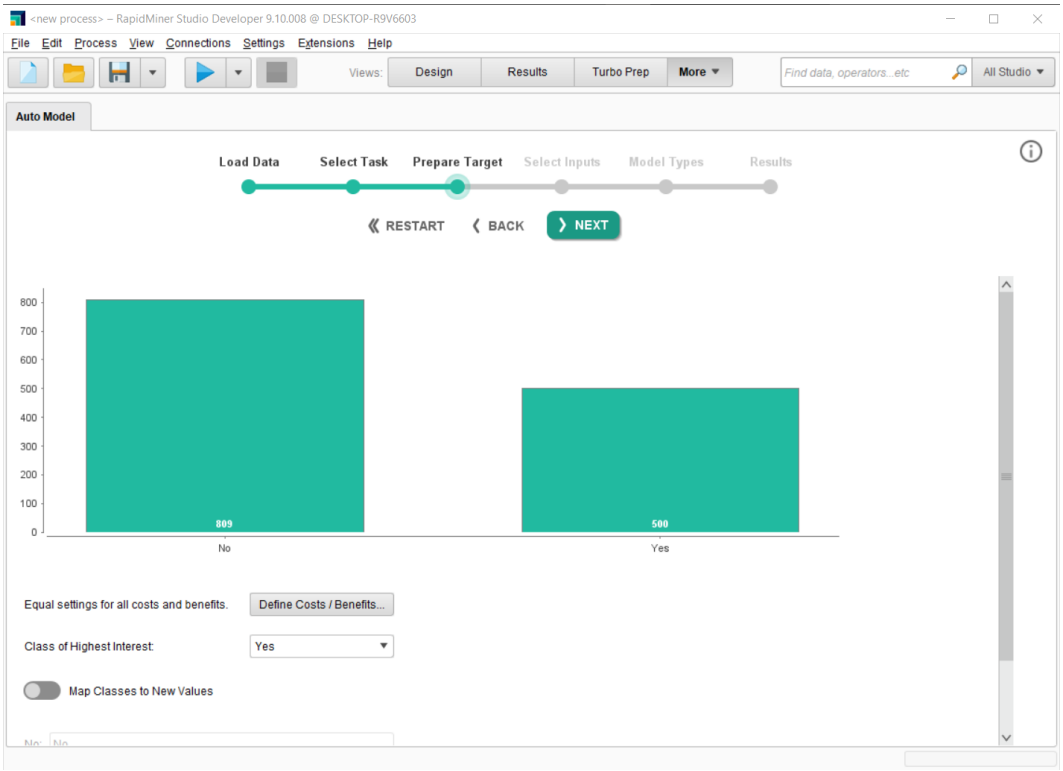


Рисунок 1.11 – Оценка балансировки

Далее необходимо выбрать входные параметры (рисунок 1.12).

Цвет статуса означает следующее:

- Красный – скорее всего, следует исключить из процесса создания модели.
- Оранжевый – необходимо перепроверить, действительно ли он нужен.
- Зеленый – важный элемент для создания модели.

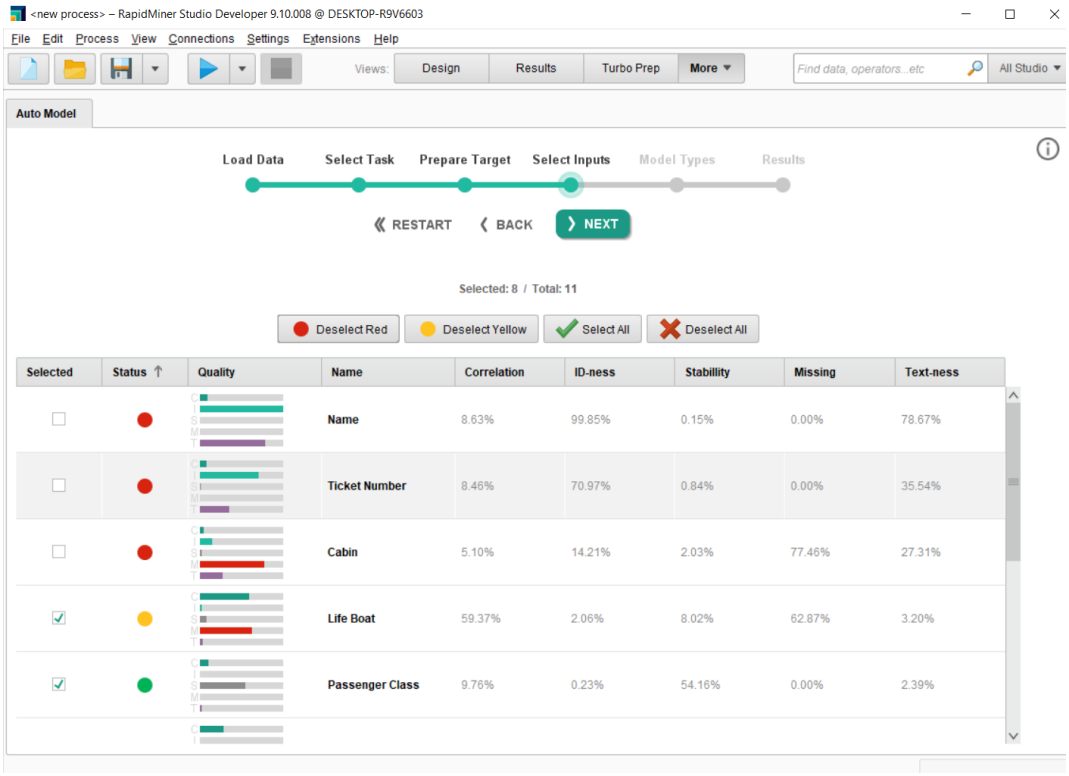


Рисунок 1.12 – Выбор входных параметров

После этого Auto Model предоставляет выбор моделей, которые имеют отношение к задаче (рисунок 1.13).

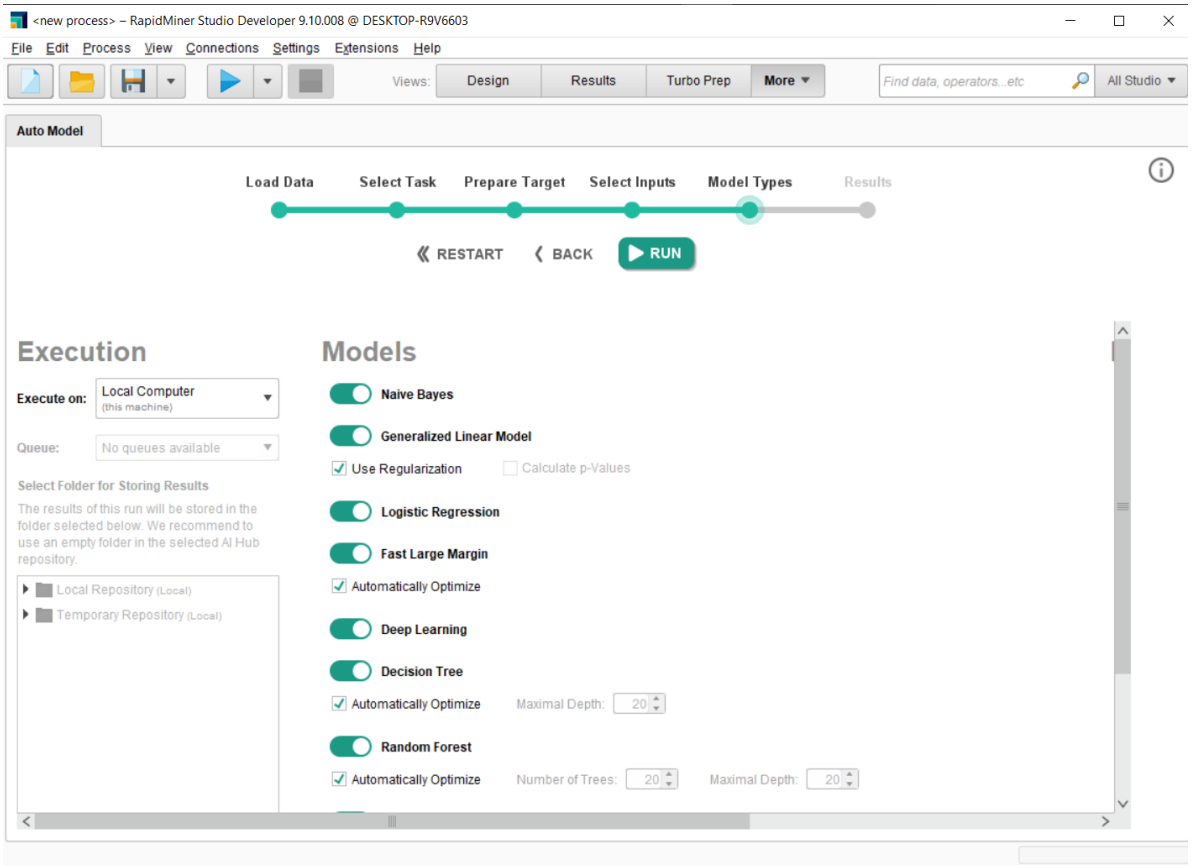


Рисунок 1.13 – Выбор моделей предсказания

Далее будет представлено краткое сравнение результатов работы моделей (рисунок 1.14).

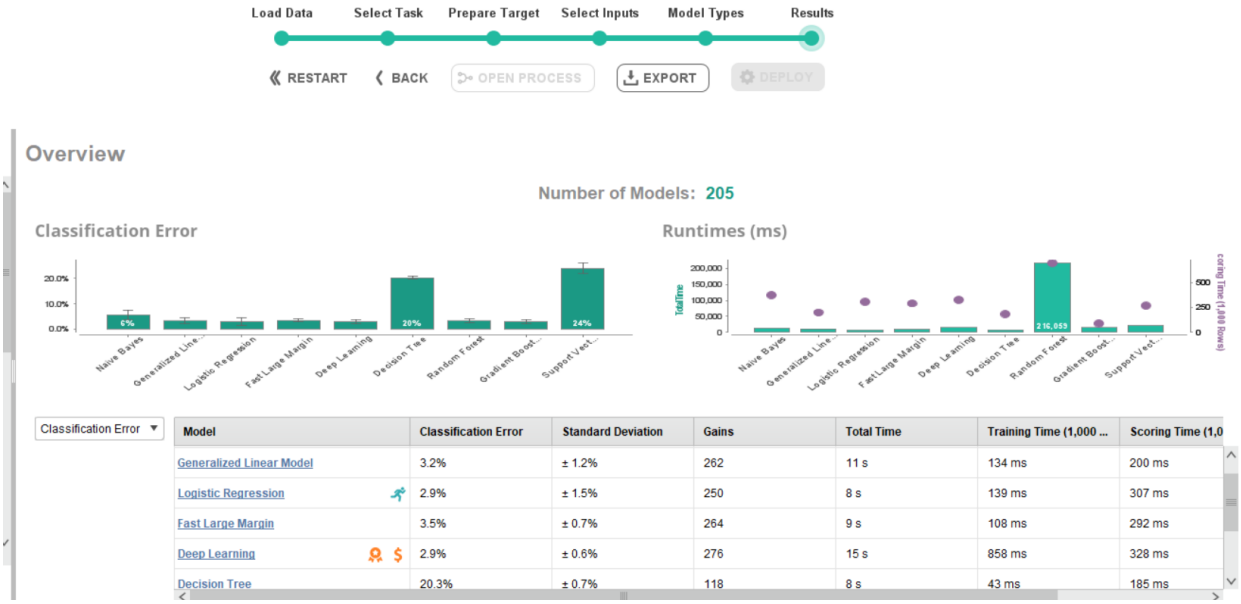


Рисунок 1.14 – Значение весов

1.3 Turbo Prep

Для того чтобы преобразовать данные необходимо перейти в раздел Transform (рисунок 1.15). В данном разделе можно провести переименование, копирование (рисунок 1.16) и удаление (рисунок 1.17), сортировку, изменение типа данных.

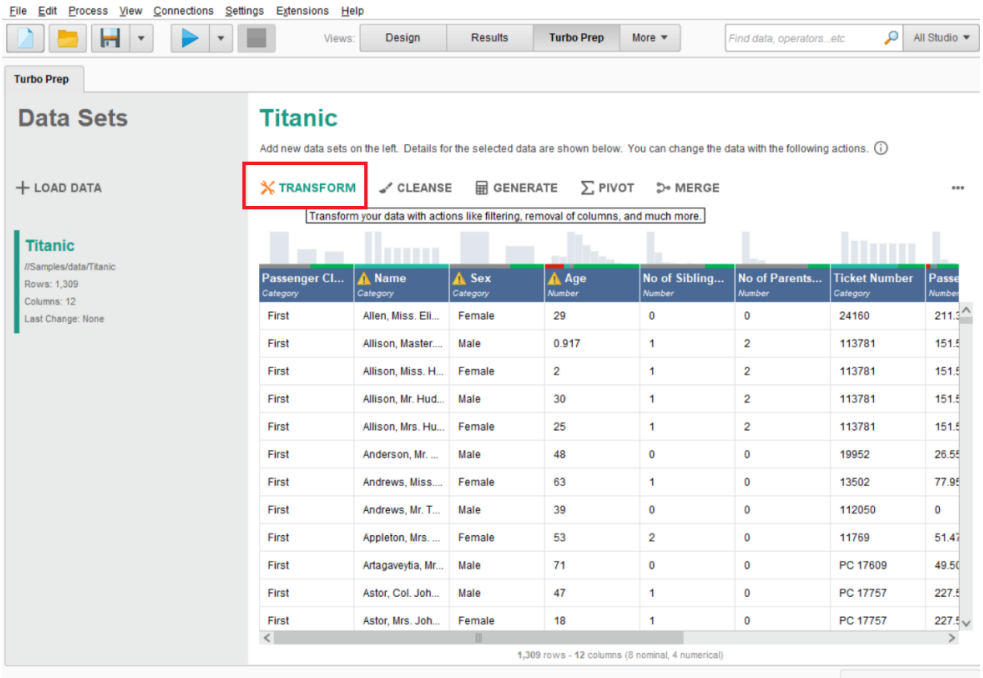


Рисунок 1.15 – Раздел Transform

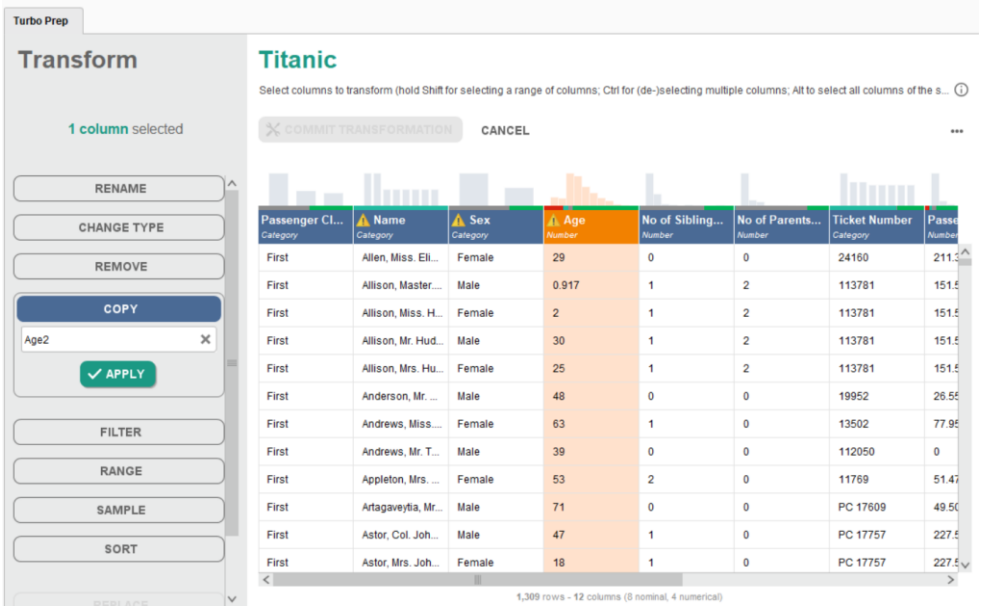


Рисунок 1.16 – Копирование атрибута

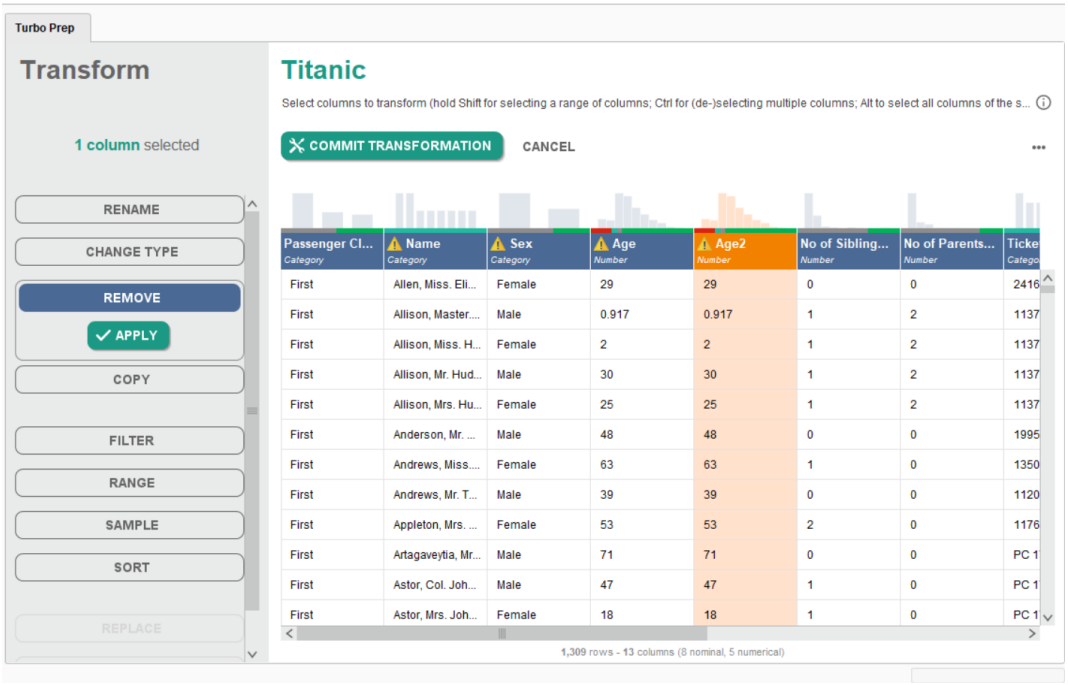


Рисунок 1.17 – Удаление атрибута

Для того чтобы провести автоматическую очистку данных необходимо перейти в раздел Cleanse (рисунок 1.18), нажать кнопку «Auto Cleansing».

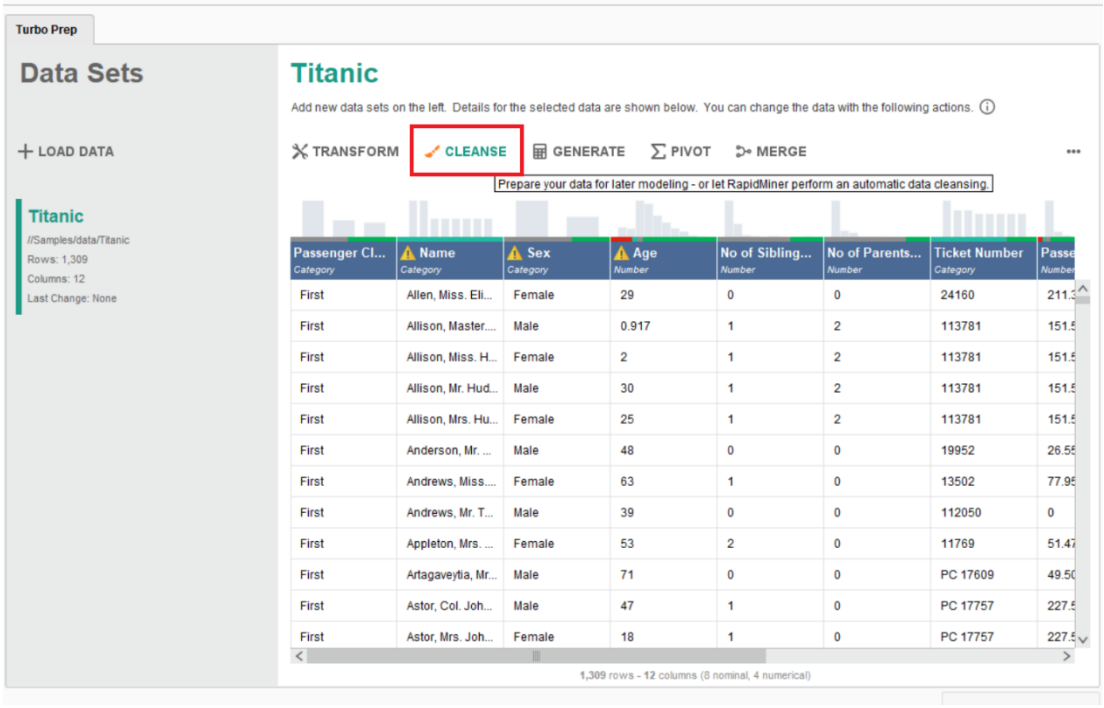
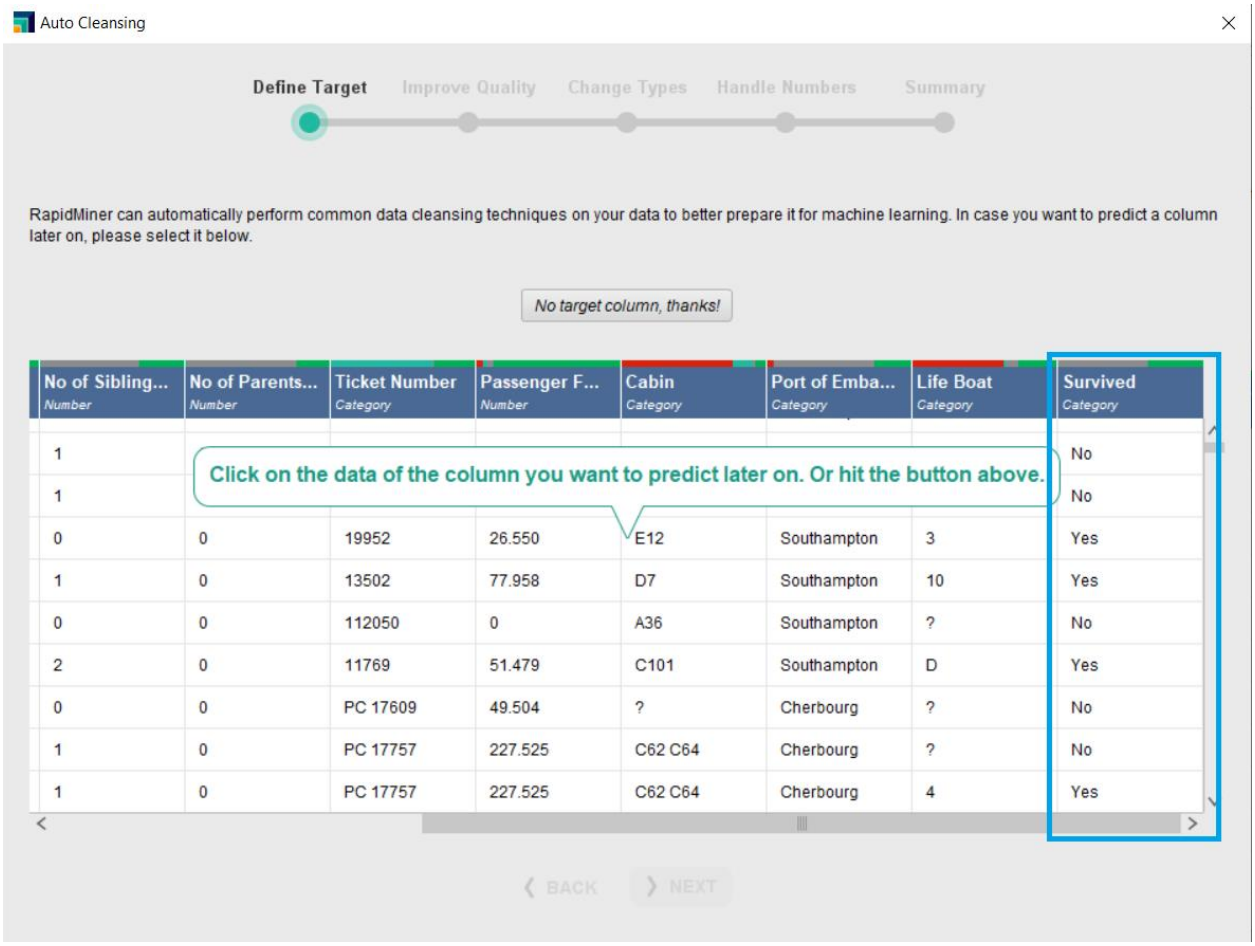


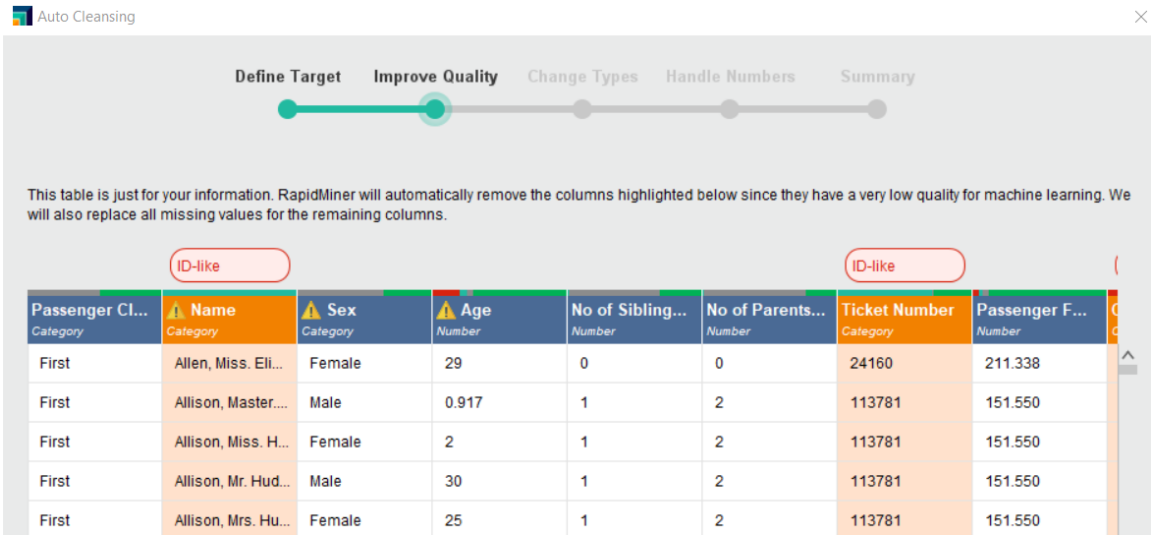
Рисунок 1.18 – Раздел Cleanse

Далее выбрать столбец, в котором записан класс (рисунок 1.19), после чего будут предложены столбцы для удаления (рисунок 1.20).



No of Sibling... Number	No of Parents... Number	Ticket Number Category	Passenger F... Number	Cabin Category	Port of Emba... Category	Life Boat Category	Survived Category
1							No
1							No
0	0	19952	26.550	E12	Southampton	3	Yes
1	0	13502	77.958	D7	Southampton	10	Yes
0	0	112050	0	A36	Southampton	?	No
2	0	11769	51.479	C101	Southampton	D	Yes
0	0	PC 17609	49.504	?	Cherbourg	?	No
1	0	PC 17757	227.525	C62 C64	Cherbourg	?	No
1	0	PC 17757	227.525	C62 C64	Cherbourg	4	Yes

Рисунок 1.19 – Выбор выходного столбца



Passenger Cl... Category	Name Category	Sex Category	Age Number	No of Sibling... Number	No of Parents... Number	Ticket Number Category	Passenger F... Number
First	Allen, Miss. Eli...	Female	29	0	0	24160	211.338
First	Allison, Master....	Male	0.917	1	2	113781	151.550
First	Allison, Miss. H...	Female	2	1	2	113781	151.550
First	Allison, Mr. Hud...	Male	30	1	2	113781	151.550
First	Allison, Mrs. Hu...	Female	25	1	2	113781	151.550

Рисунок 1.20 – Удаление столбцов

На следующем шаге будет предложено изменить все типы столбцов на числовые или категориальные, здесь следует нажать «Next» (рисунок 1.21).

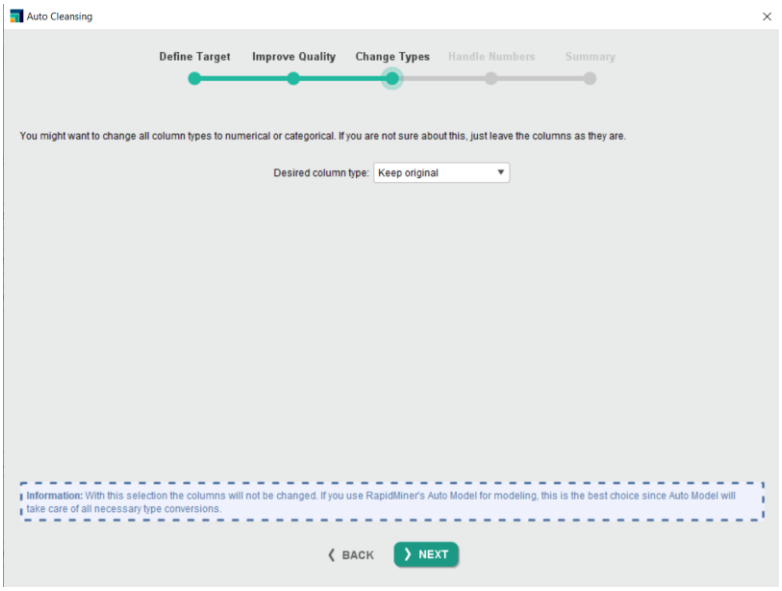


Рисунок 1.21 – Изменение типа данных

После этого будет предложено дополнительно выбрать провести анализ основных компонентов и/или нормализацию (рисунок 1.22), здесь стоит нажать «Next» и применить все настройки.

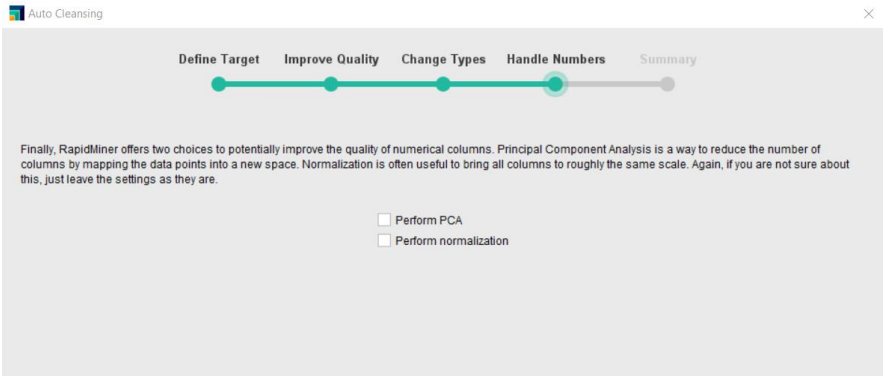


Рисунок 1.22 – Обработка чисел

Для генерации новых столбцов нужно перейти в раздел Generate (рисунок 1.23), выбрать данные и действия, которые будут над ними совершаться (рисунок 1.24).

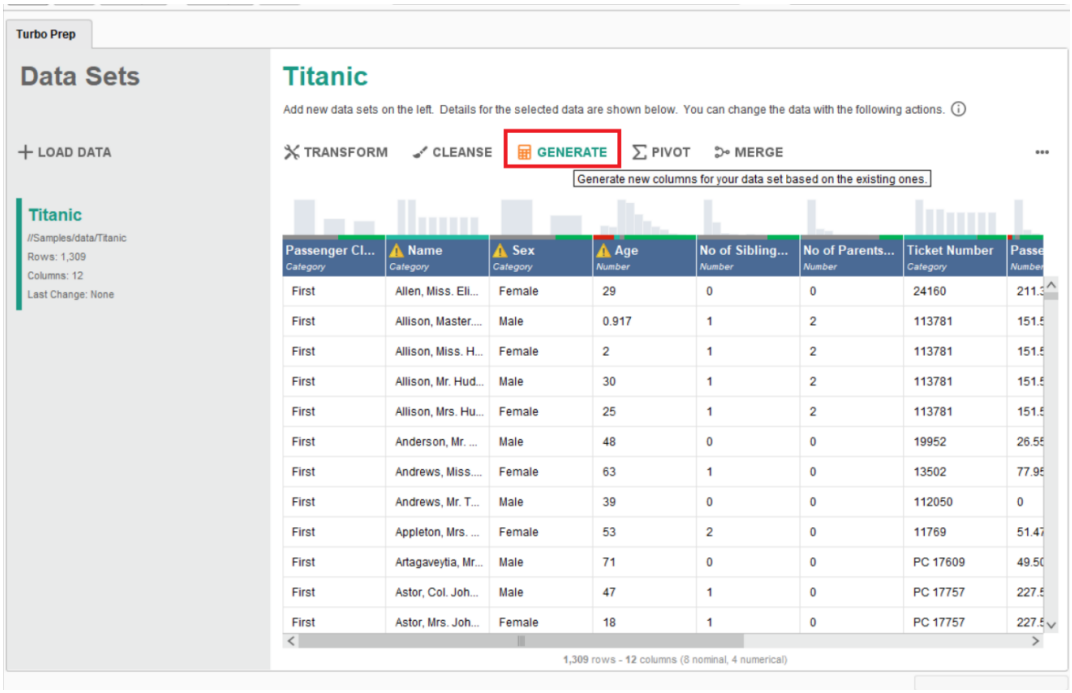


Рисунок 1.23 – Раздел Generate

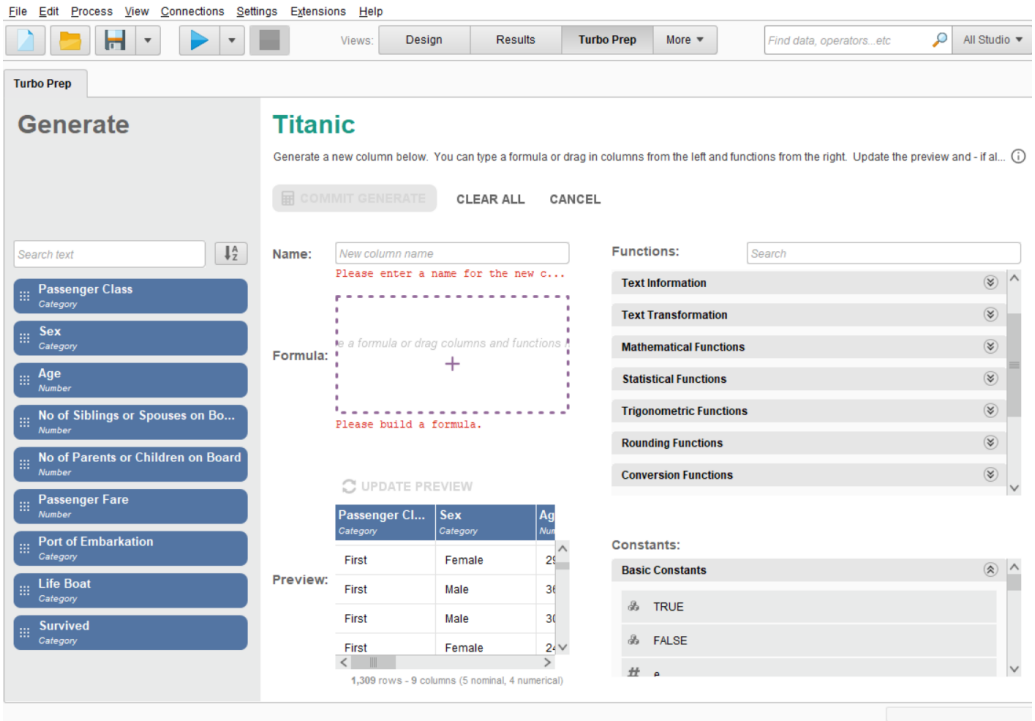


Рисунок 1.24 – Генерация нового столбца

Раздел Pivot (рисунок 1.25) позволяет решить задачу создания сводных таблиц. Для этого в левое поле необходимо перенести столбец, значения которого будут использоваться в качестве идентификатора, в верхнее поле столбец, значения которого будут использоваться в качестве заголовков, в

нижнее - столбцы, значения которых будут отображены в сводном виде (рисунок 1.26).

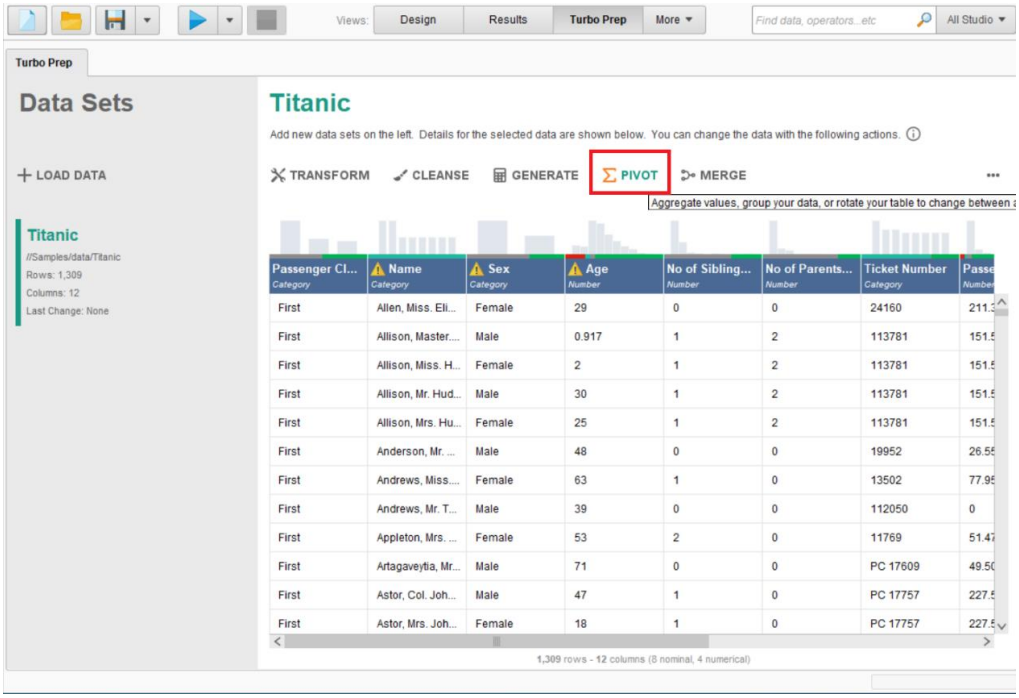


Рисунок 1.25 – Раздел Pivot

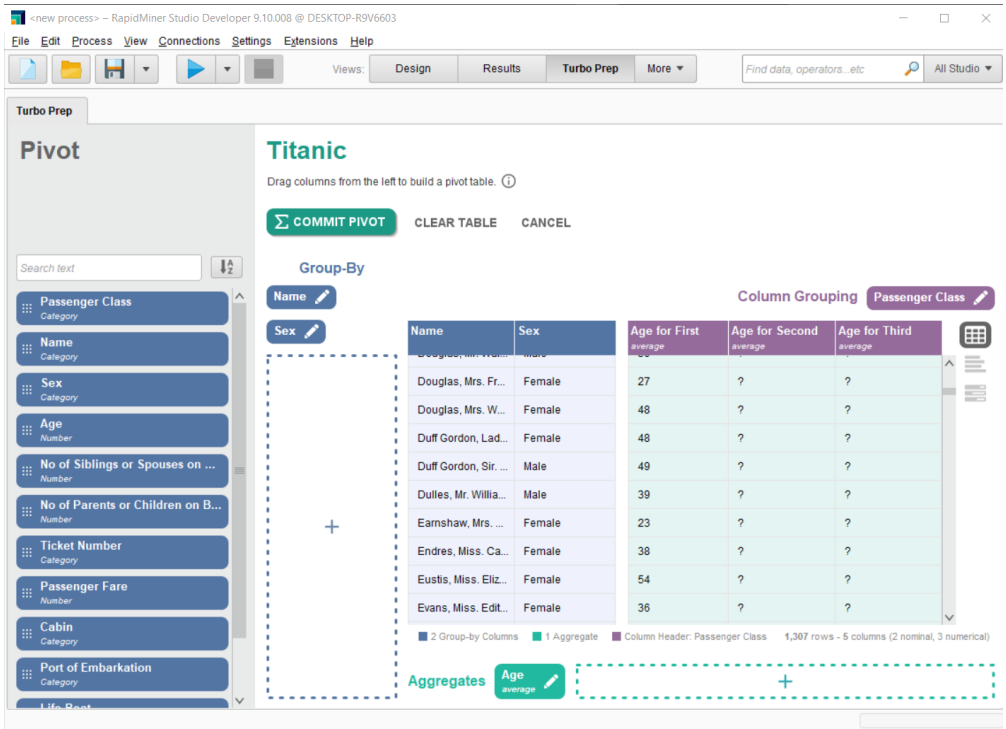


Рисунок 1.26 – Создание новой таблицы

2 Задание на лабораторную работу.

1. Загрузить набор данных согласно варианту.
2. Выполнить ход работы.
3. Проанализировать структуру данных, включая количество объектов, атрибутов и классов, статистику по всем атрибутам, а также привести пример визуализации данных.
4. Использовать Auto Model для разработки модели классификации.
5. Проанализировать результаты классификации каждой модели, привести значения метрик, распределение весов для каждого атрибута, а также процесс для одной из модели.
6. Перейти в раздел Transform и провести преобразование данных: переименование, копирование и удаление, сортировку, изменение типа данных.
7. Перейти в раздел Cleanse и провести автоматическую очистку данных, провести нормализацию данных.
8. Перейти в раздел Generate и провести генерацию новых столбцов.
9. Перейти в раздел Pivot и провести создание сводной таблицы.
10. Повторить пункт 3 задания на основе измененных данных. После сравнить результаты, полученные с помощью построенной модели, с результатами, полученными на основе исходных данных до обработки.
11. Написать отчет в соответствии с ОС ТУСУР

В таблице 2.1 представлены варианты индивидуального задания

Таблица 2.1 – Индивидуальное задание

Вариант	Набор данных
1	Lung Cancer Dataset (kaggle.com)
2	Heart Disease Dataset (kaggle.com)
3	Diabetes Dataset (kaggle.com)
4	Pima Indians Diabetes Database (kaggle.com)
5	Fetal Health Classification (kaggle.com)

Контрольные вопросы

1. Что такое RapidMiner и для чего он используется?
2. Какие форматы данных поддерживает RapidMiner?
3. Какие шаги необходимо выполнить для импорта данных?
4. Какие инструменты находятся в разделе Turbo Prep и для чего они используются?
5. Какие типы задач можно решать с помощью Auto Model?