Отчет 2

Цель задания:

Цель данного исследования - построить классификационную модель, позволяющую точно предсказать состояние здоровья пациента на основе анализа данных электрокардиограммы (ЭКГ). Мы использовали различные методы машинного обучения и автоматического машинного обучения (AutoML), а также интегрировали методы PSA и t-SNE. Цель - изучить эффективность различных методов при анализе данных в здравоохранении и в итоге выбрать оптимальную модель для практического применения.

Экспериментальные методы:

подготовка данных, применение t-SNE и PCA

Экспериментальная среда

colab

результат:

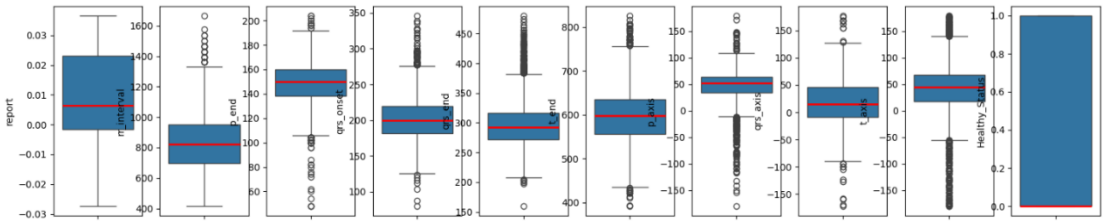

Рисунок 1

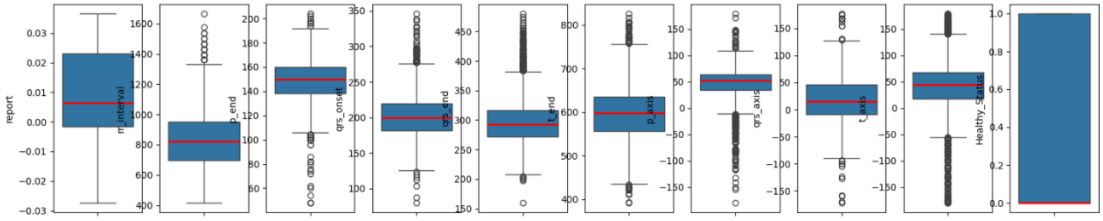

Рисунок 2

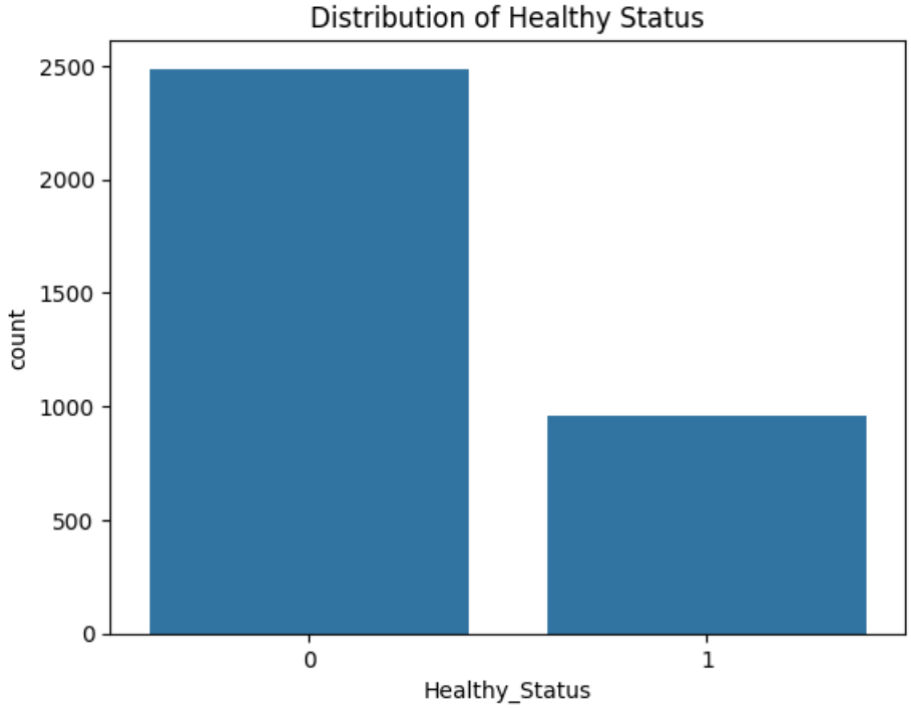

Рисунок 3

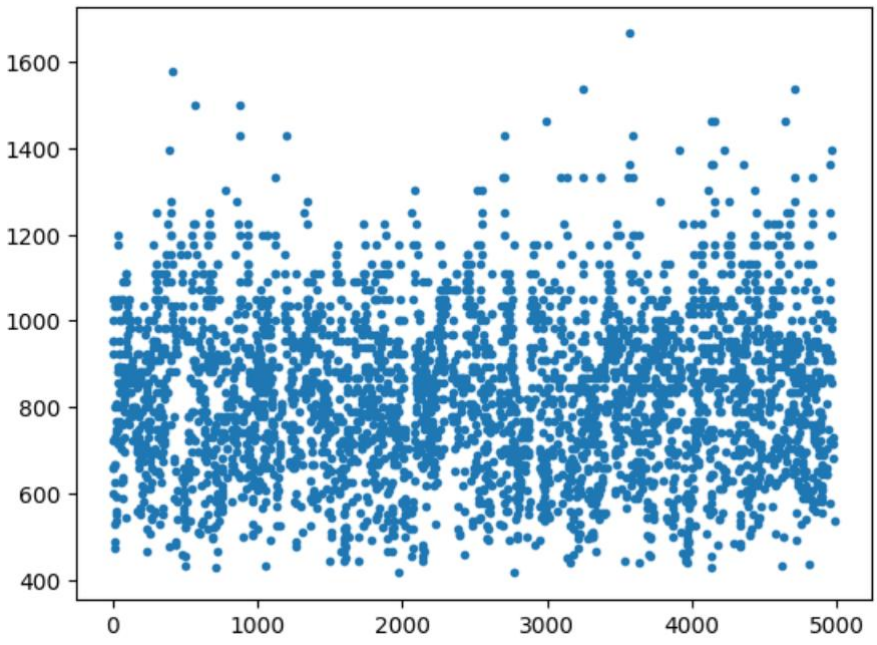

Рисунок 4

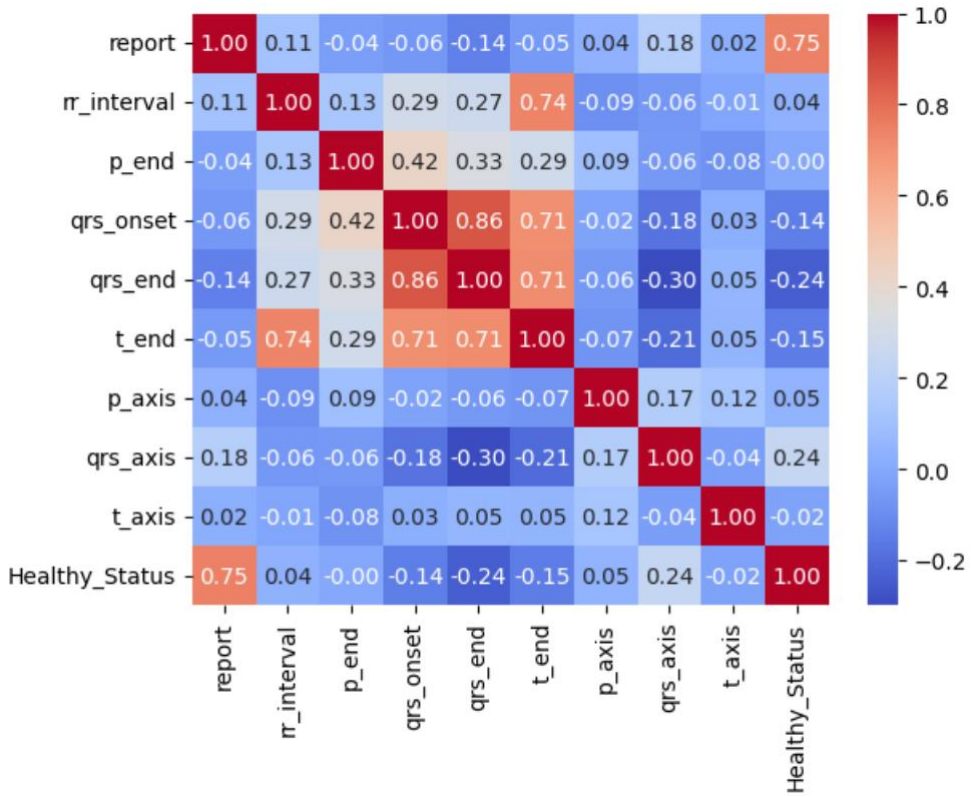

Рисунок 5

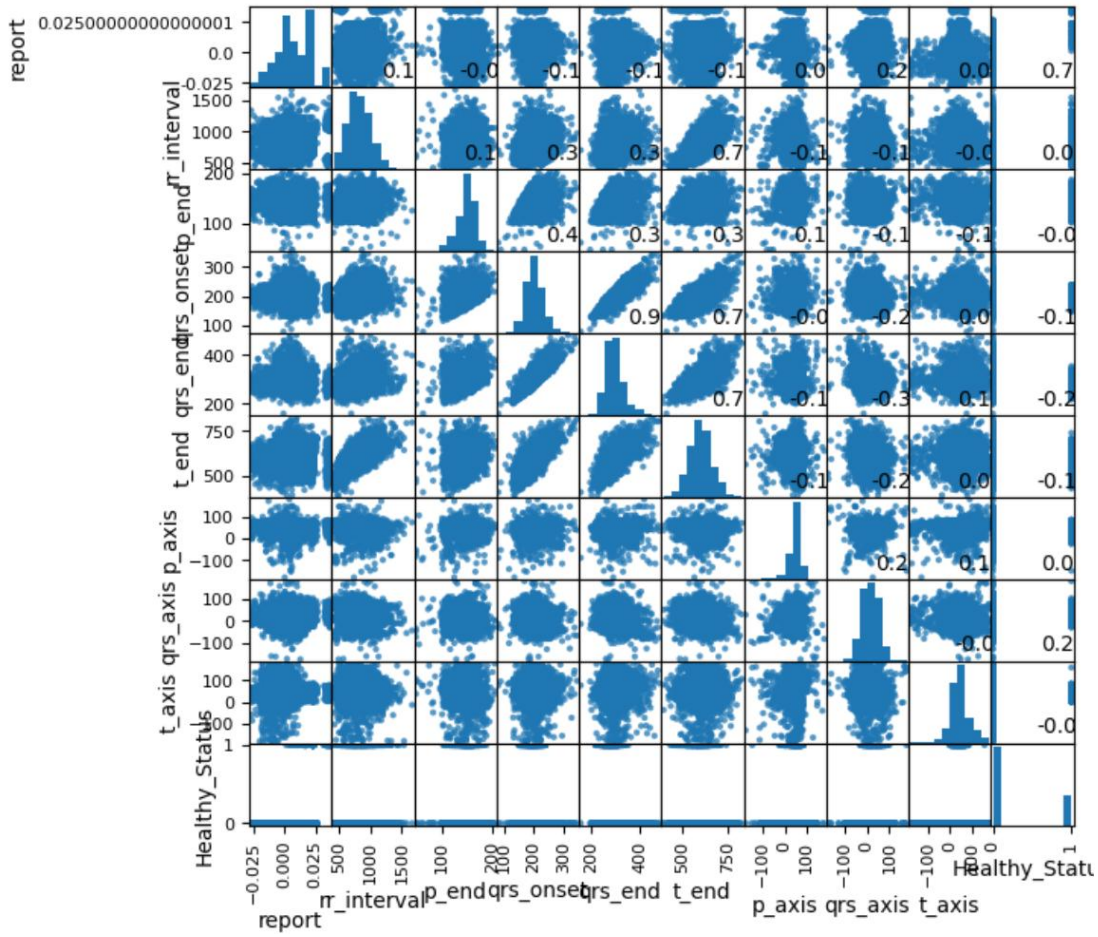

Рисунок 6

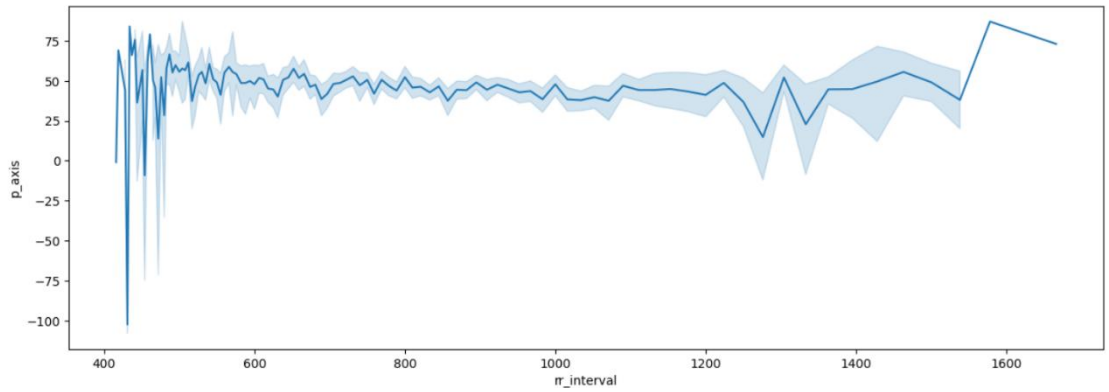

Рисунок 7

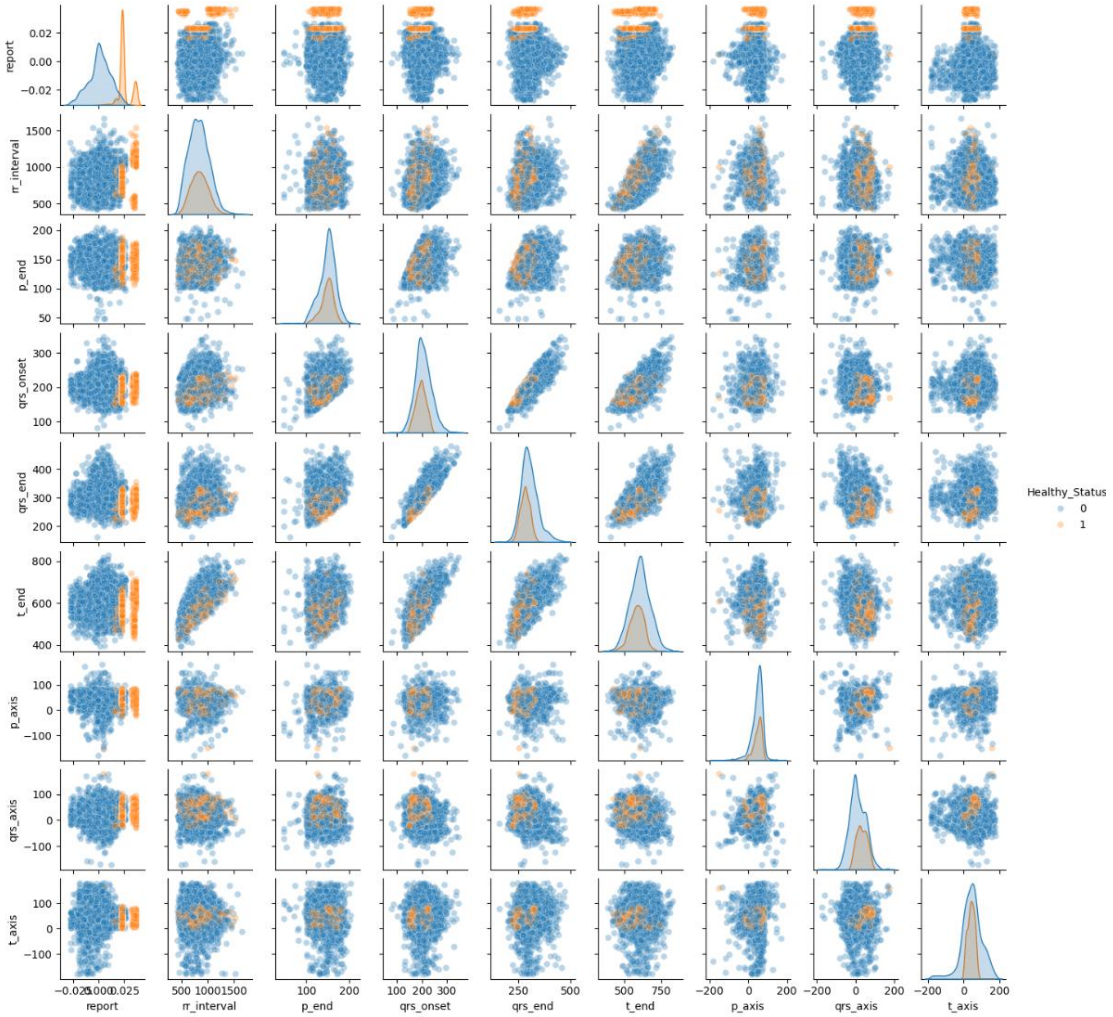

Рисунок 8

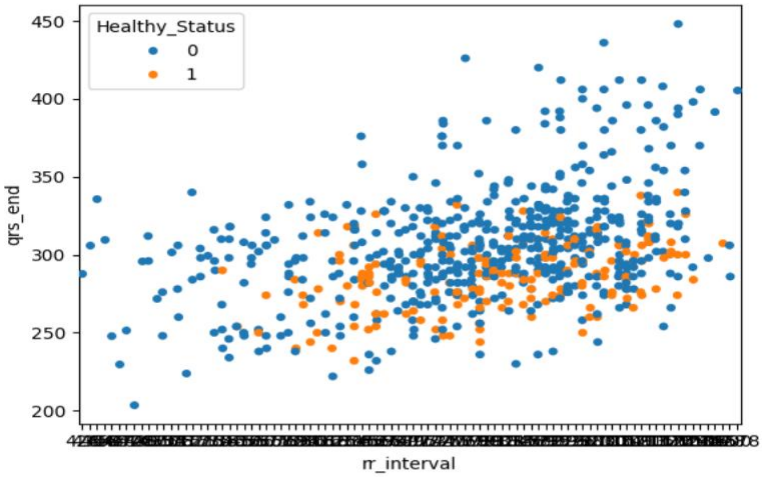

Рисунок 9

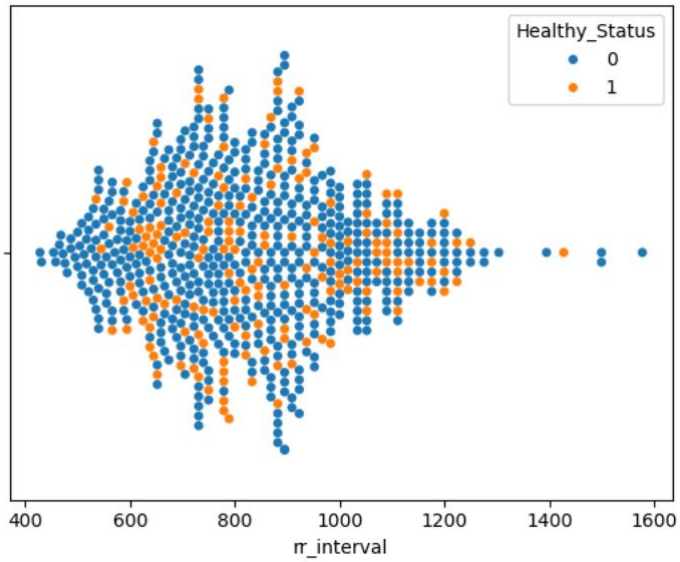

Рисунок 10

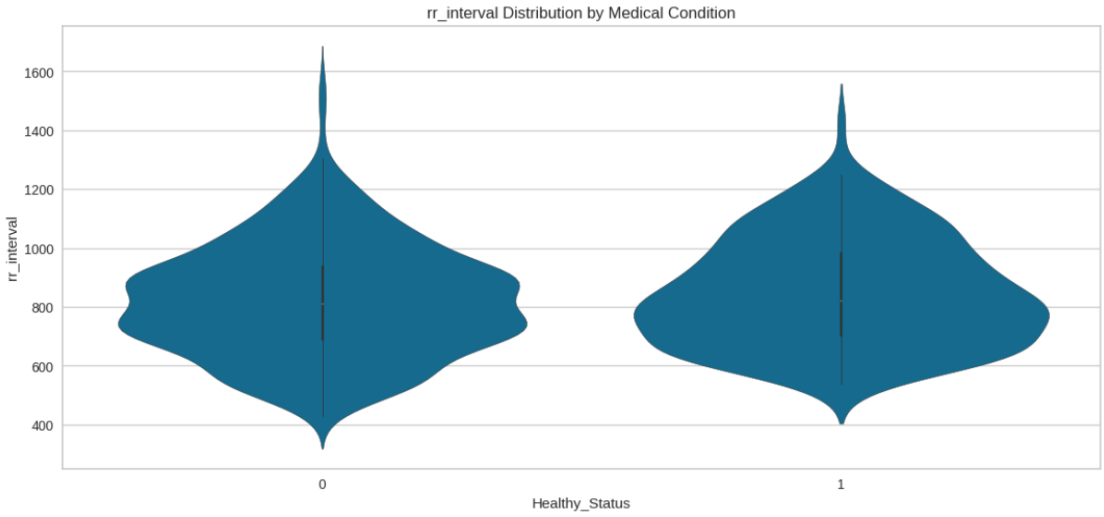

Рисунок 11

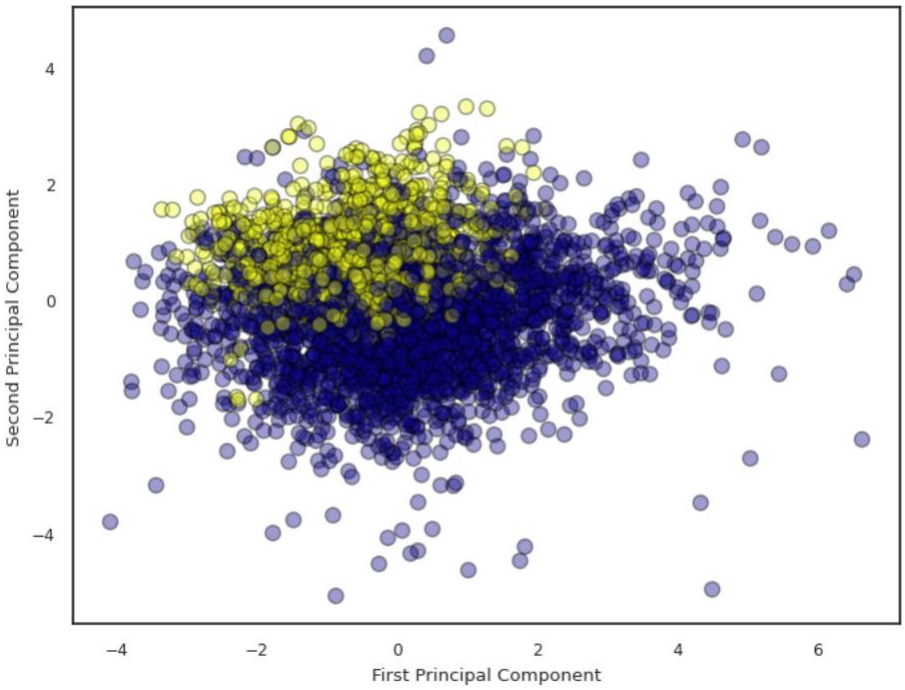

Рисунок 12

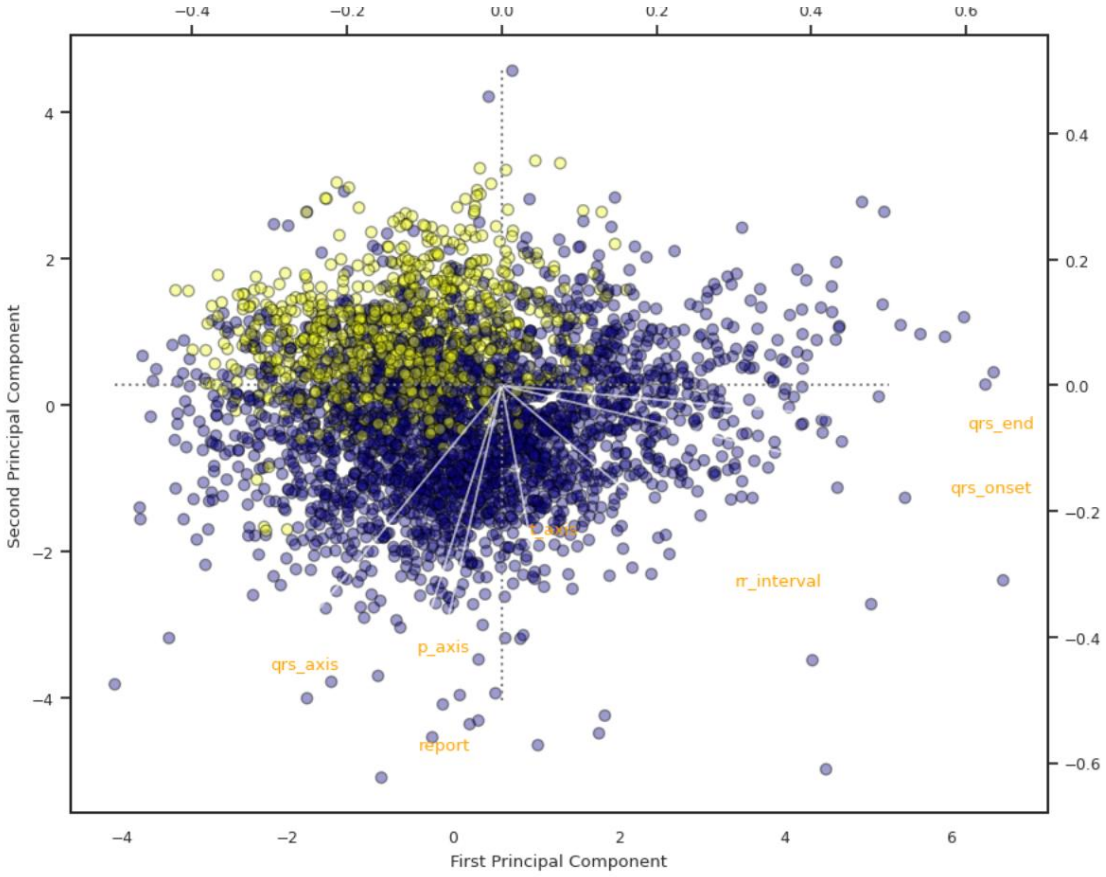

Рисунок 13

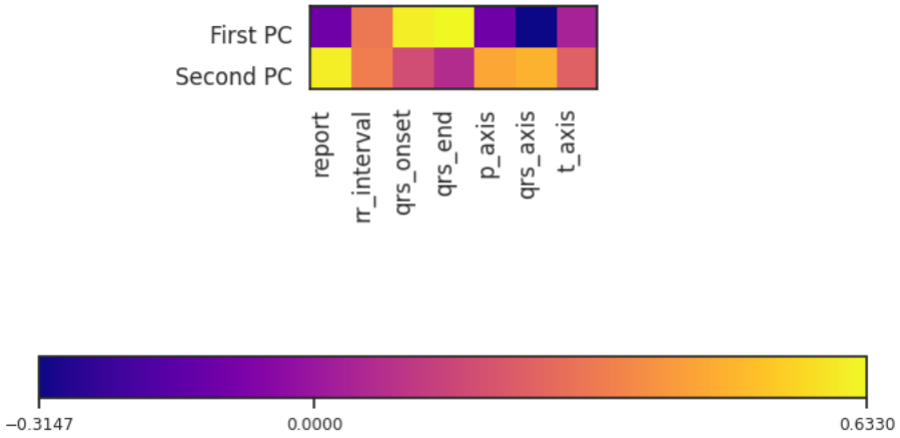

Рисунок 14

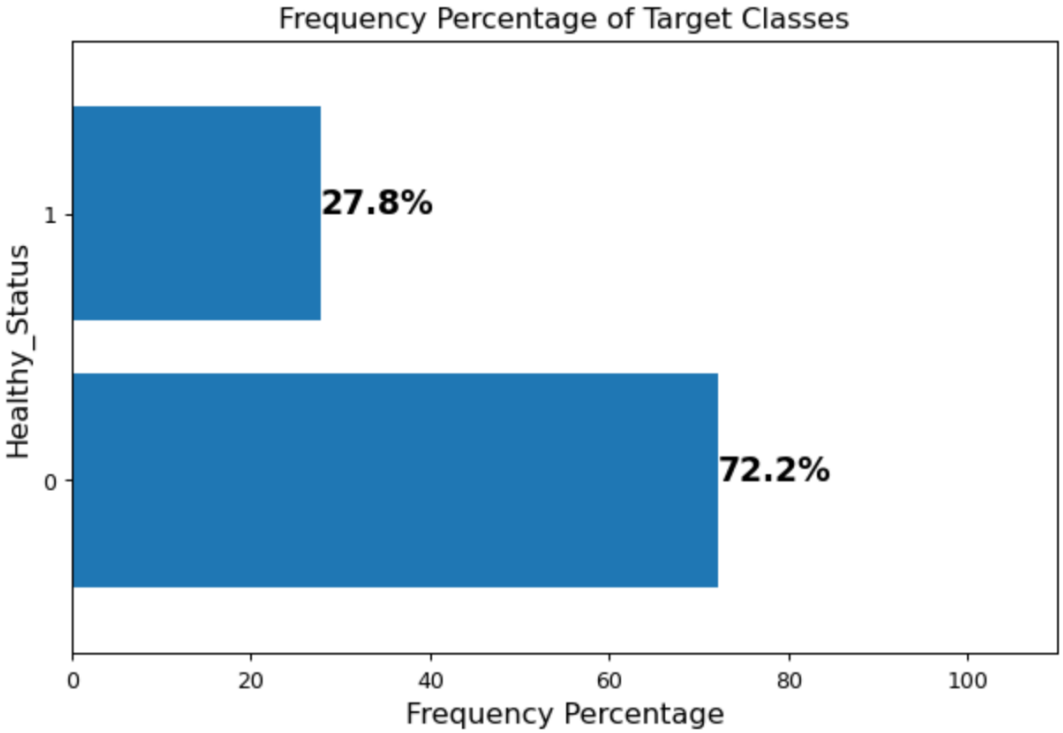

Рисунок 15

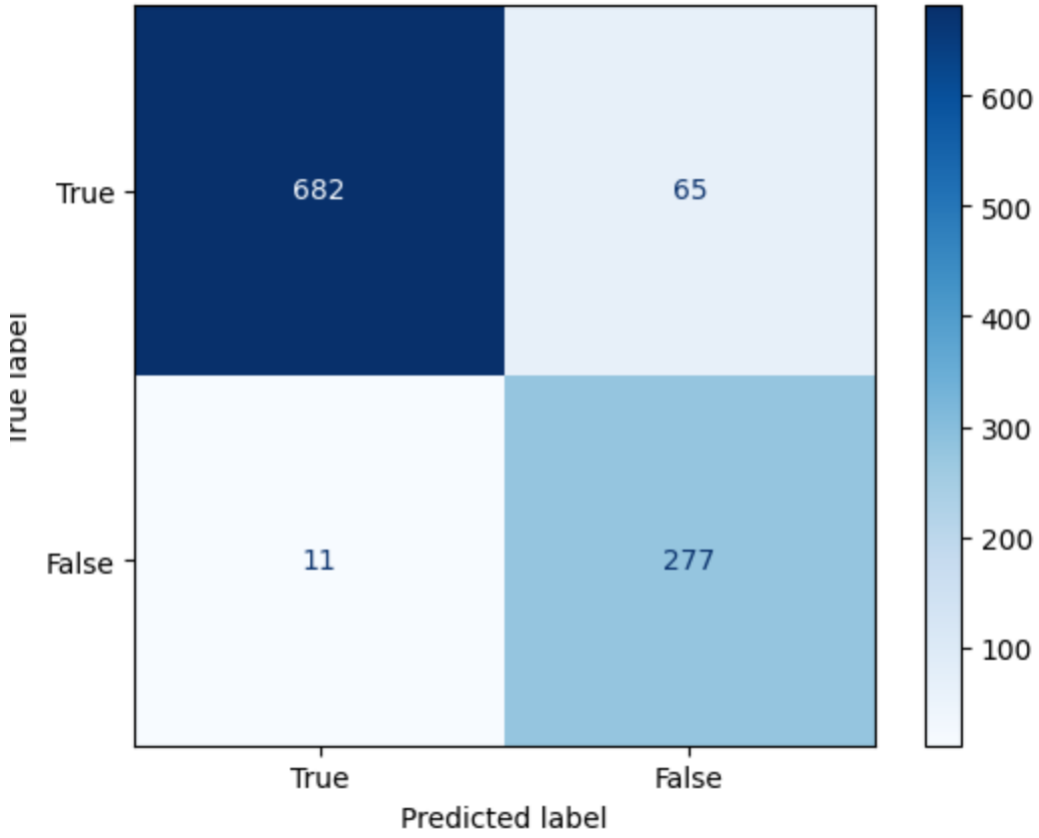

Рисунок 16

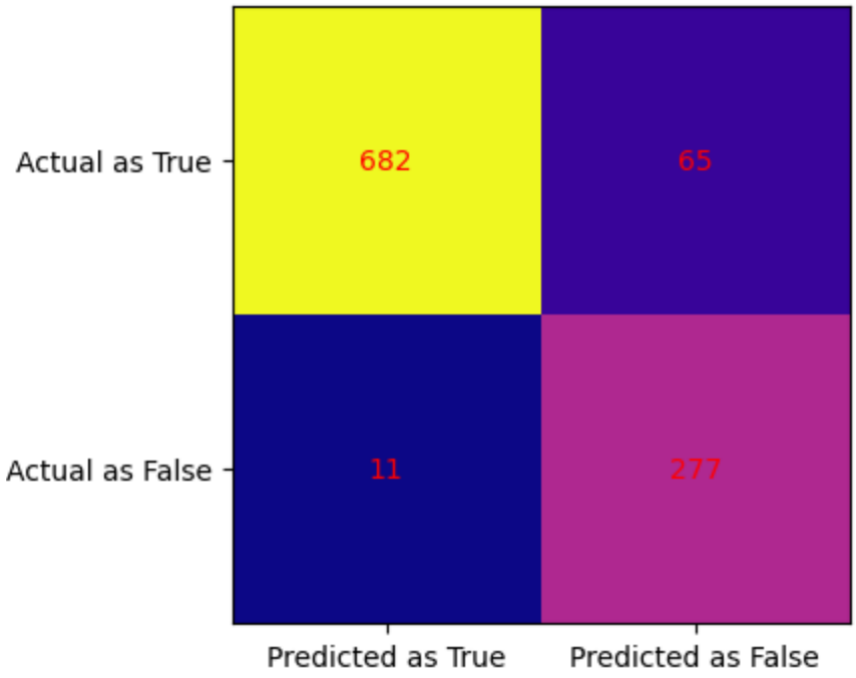

Рисунок 17

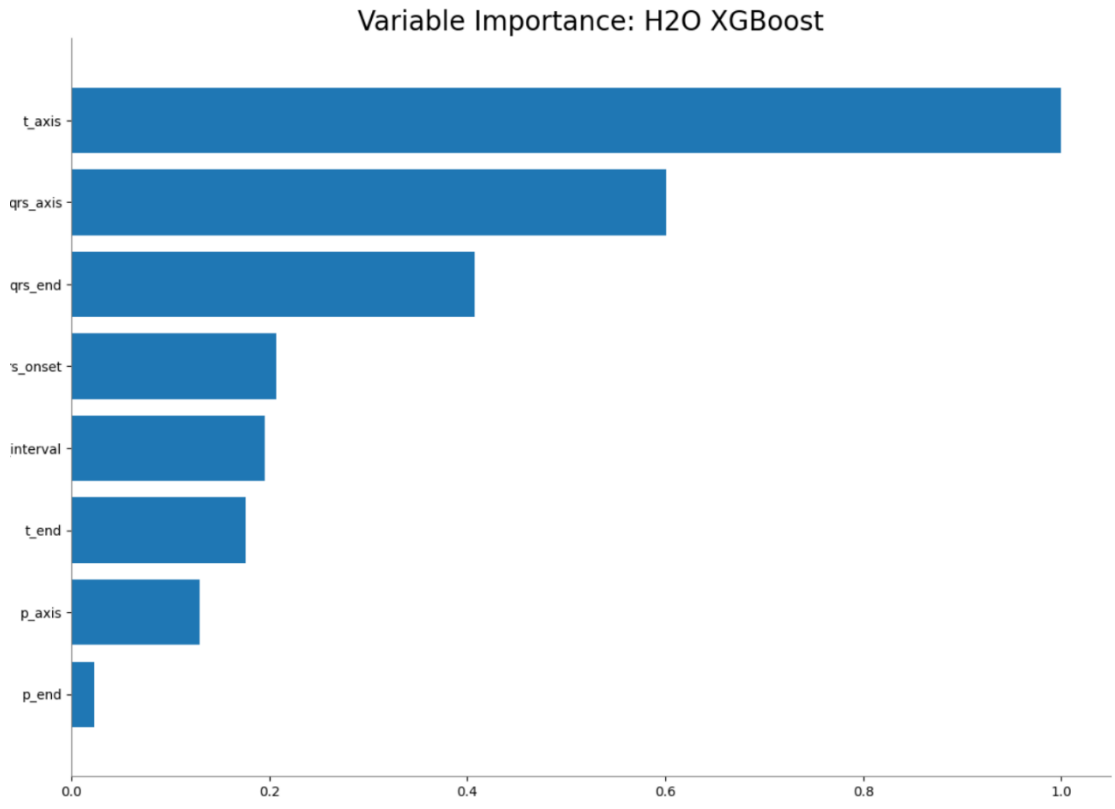
Variable Importance: H2O XGBoost
t_axis
qrs_axis
qrs_end
's_onset
interval
t_end
p_axis
p_end
0.0
0.2
0.4
0.6
0.8
1.0

Рисунок 18

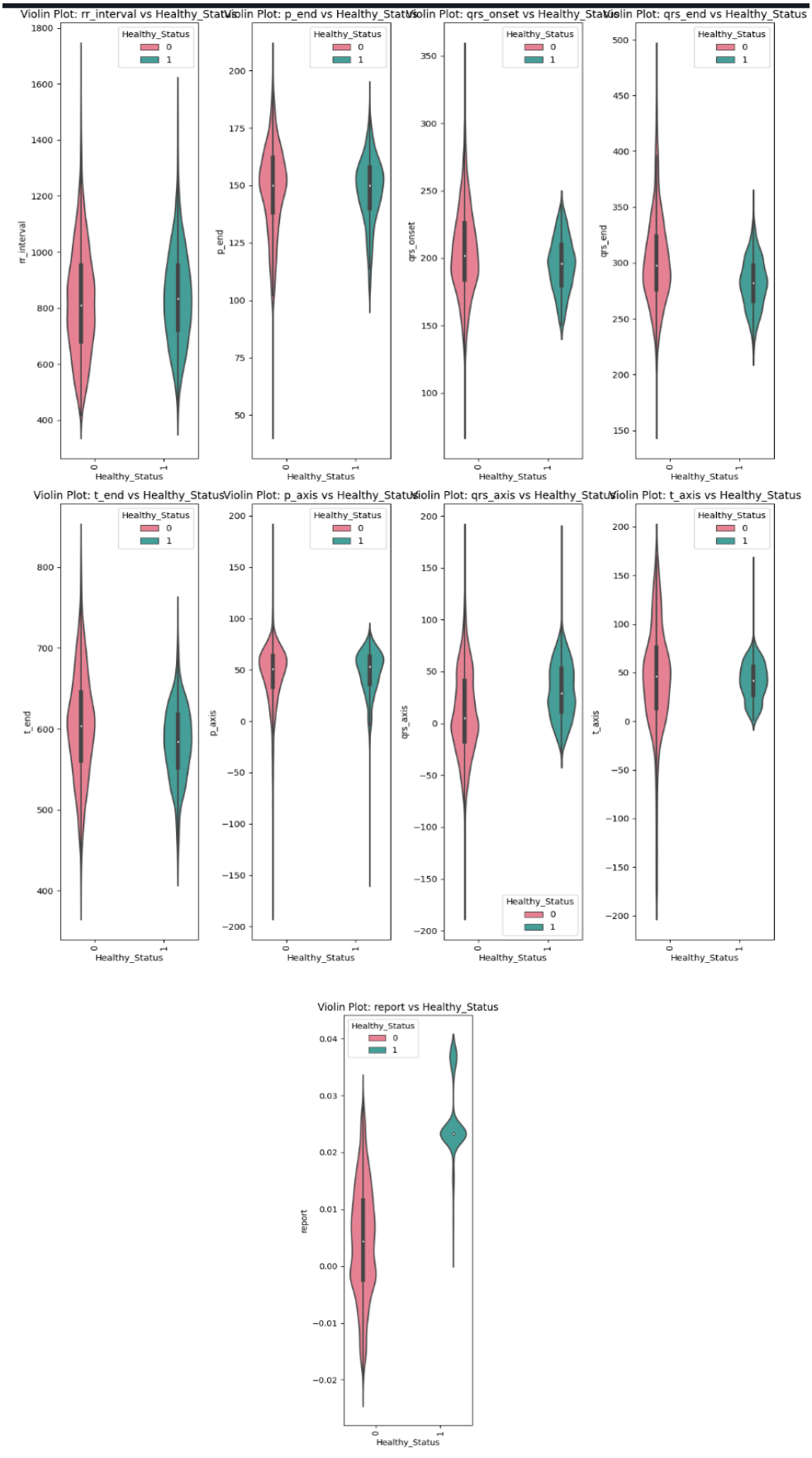
Violin Plot: rr_interval vs Healthy_Status
Violin Plot: p_end vs Healthy_Status
Violin Plot: qrs_onset vs Healthy_Status
Violin Plot: qrs_end vs Healthy_Status
Violin Plot: t_end vs Healthy_Status
Violin Plot: p_axis vs Healthy_Status
Violin Plot: qrs_axis vs Healthy_Status
Violin Plot: t_axis vs Healthy_Status
Violin Plot: report vs Healthy_Status
Healthy_Status
0
1
rr_interval
p_end
qrs_onset
qrs_end
t_end
p_axis
qrs_axis
t_axis
report

Рисунок 19

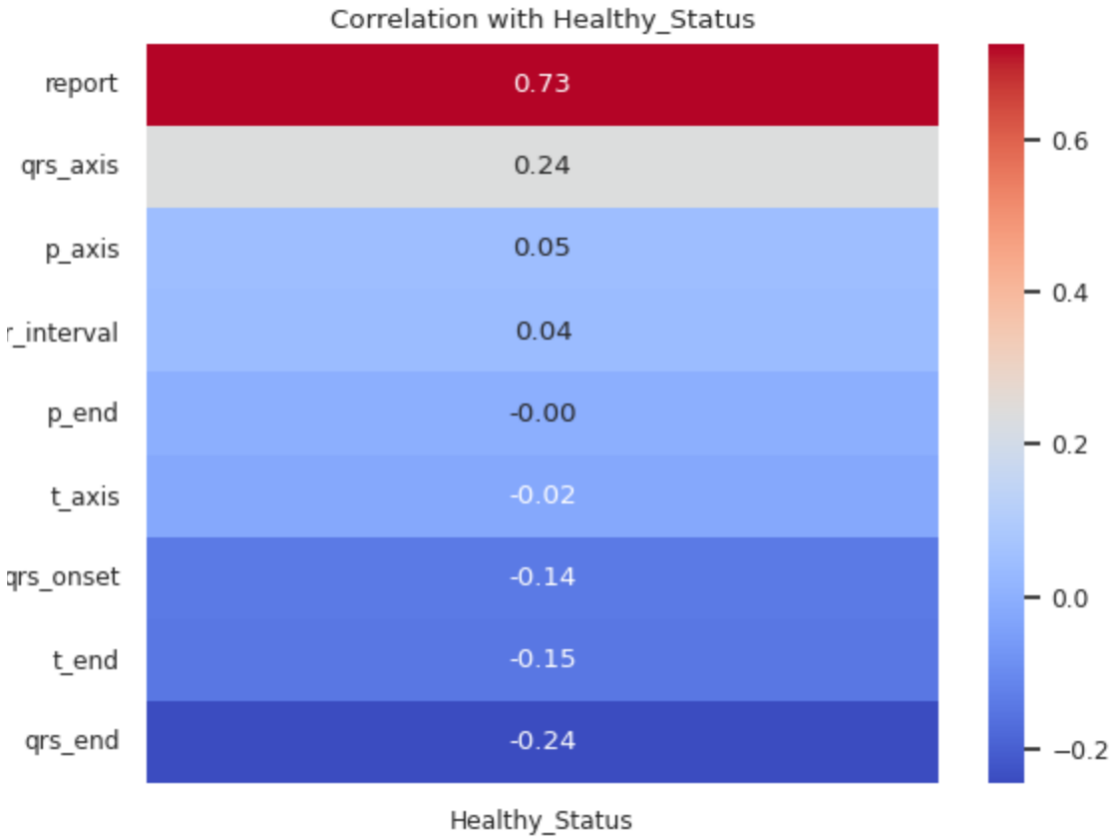
Correlation with Healthy_Status
report
qrs_axis
p_axis
r_interval
p_end
t_axis
qrs_onset
t_end
qrs_end
0.73
0.24
0.05
0.04
-0.00
-0.02
-0.14
-0.15
-0.24
0.6
0.4
0.2
0.0
−0.2
Healthy_Status

Рисунок 20

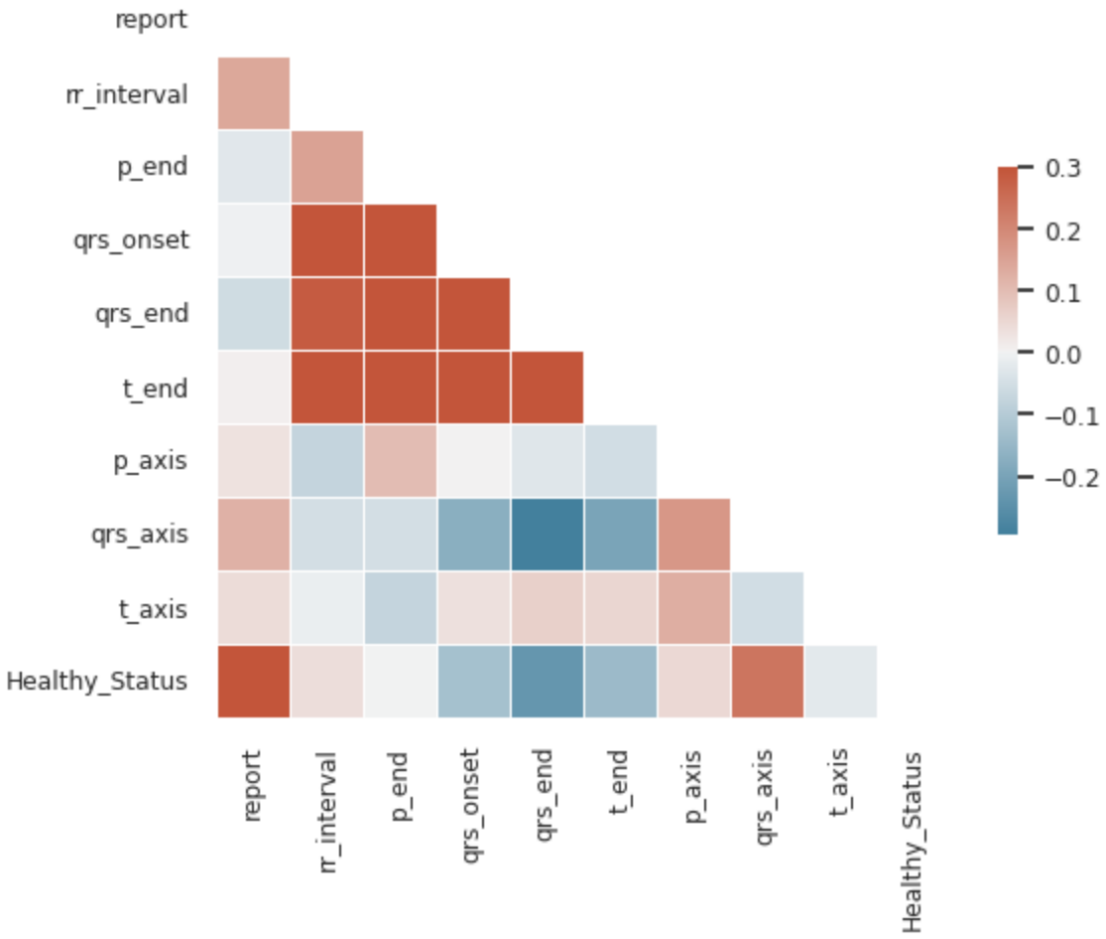
report
rr_interval
p_end
qrs_onset
qrs_end
t_end
p_axis
qrs_axis
t_axis
Healthy_Status
0.3
0.2
0.1
0.0
−0.1
−0.2

Рисунок 21

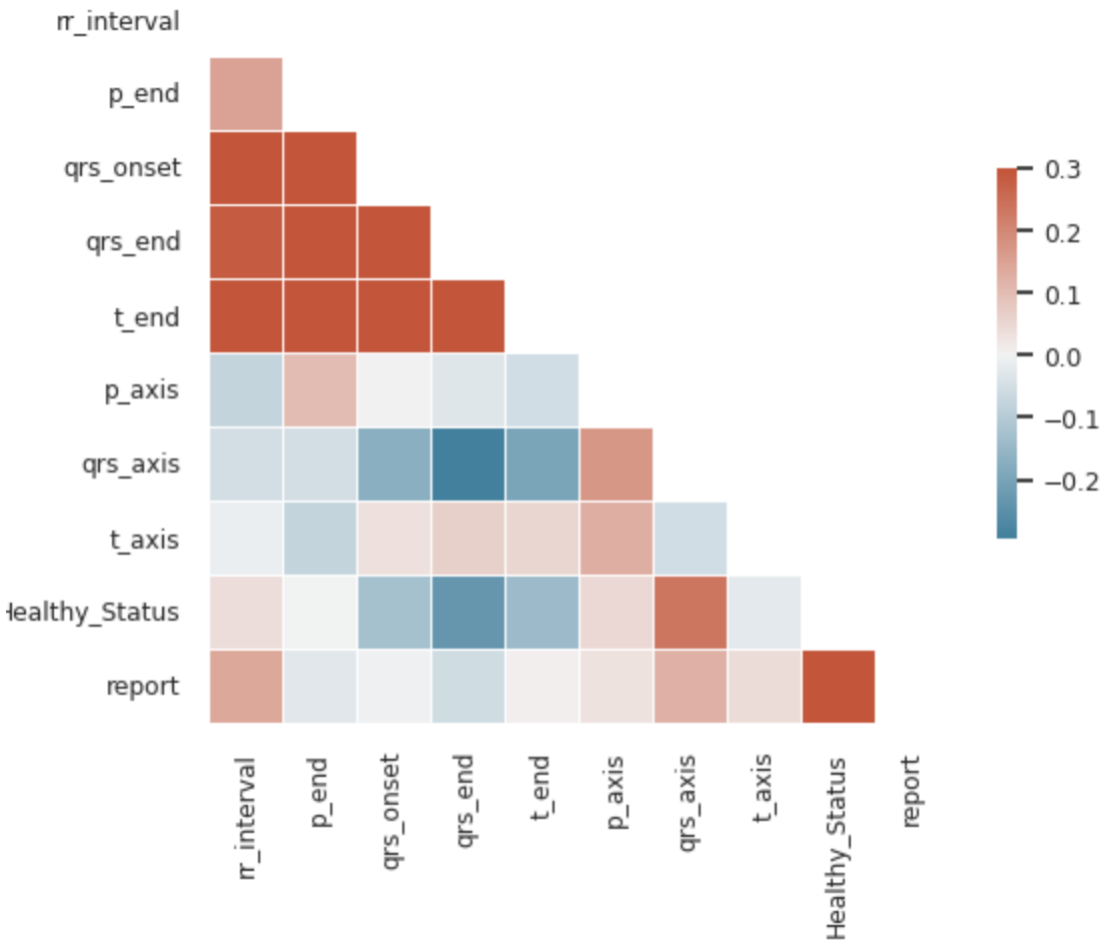
rr_interval
p_end
qrs_onset
qrs_end
t_end
p_axis
qrs_axis
t_axis
Healthy_Status
report
0.3
0.2
0.1
0.0
−0.1
−0.2

Рисунок 22

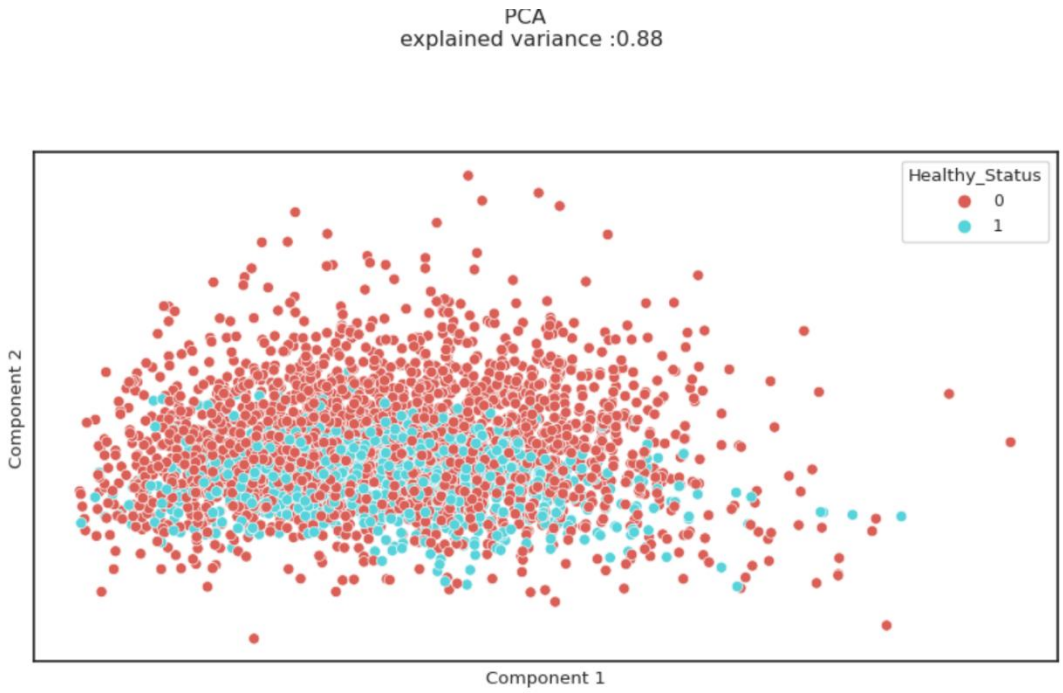

Рисунок 23

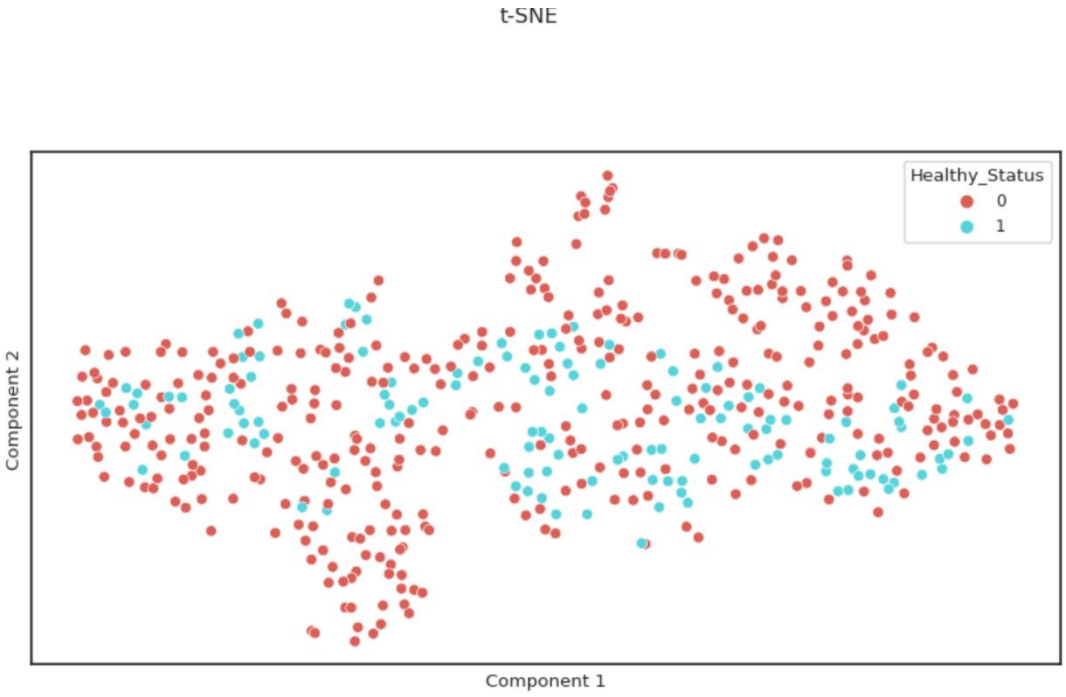

Рисунок 24

заключение

1. предварительная обработка данных

Обработка выбросов: фильтрация экстремальных значений (например, rr_interva1 >2000) для обеспечения обоснованности числовых характеристик.0

Обработка текстовых признаков: объединение 18 текстовых отчетов в одно поле, генерация средних векторов слов в предложениях с помощью модели Word2Vec и сжатие их в скалярные признаки.

Очистка данных:Удаление избыточных столбцов (например, пропускная способность, фильтрация), исправление ошибок в названиях столбцов (eeg_time и eeg_date).

2. эксплораторный анализ данных (EDA)

Визуализация распределений: квадратные и линейные графики, графики скрипок для отображения распределения признаков и различий между категориями (например, распределение rr_interva1 в здоровых и нездоровых группах).

распределение в группах здоровых и нездоровых)

Корреляционный анализ: тепловые карты и матрицы рассеяния выявляют линейные связи между признаками (например, сильная положительная корреляция между осями p_axis и qrs_axis).

Анализ снижения размерности: PCA сжимает многомерные

признаки до 2 главных компонент (с суммарной объясненной дисперсией около 64,5 %), визуализируя э)циальную разделимость классов в пространстве главных компонент.

3.Построение и оптимизация моделей

Базовая модель:В качестве базовой модели используется гауссовский парк Байеса (GaussianNB), точность тестового набора составляет около 72,8%.0

Оптимизация AutoML:0

:H20 AutoML:Автоматическое обучение нескольких моделей (например, XGBoost, GBM) за 60 секунд, точность оптимальной модели на тестовом наборе - 80,2%, оценка F1 - 0,81.

:AutoGluon:Дальнейшее повышение производительности за счет интегрированного обучения, точность увеличилась до 82,5%. Анализ интерпретируемости:График важности признаков XGBoost показывает, что rr_interva1 и qrs_axis являются ключевыми предикторами. Валидация и оценка4

Матрица запутанности:Сравните предсказанные и истинные метки и рассчитайте Recall, Precision и F1 score.

Визуализация границ принятия решений:Продемонстрируйте область принятия решений классификатора в двумерном пространстве с помощью снижения размерности PCA, чтобы визуально проверить делимость модели.

основной алгоритм

```
# ======================
# 1. 基础库导入 / Импорт базовых библиотек
# ======================
import numpy as np  # 数值计算库 / Библиотека для численных вычислений
import pandas as pd  # 数据处理库 / Библиотека для обработки данных
import matplotlib.pyplot as plt  # 可视化库 / Библиотека визуализации
import seaborn as sns  # 高级可视化库 / Продвинутая библиотека визуализации
from sklearn.naive_bayes import GaussianNB  # 高斯朴素贝叶斯分类器 / Наивный байесовский классификатор
from sklearn.metrics import accuracy_score  # 准确率评估 / Метрика точности
from sklearn.preprocessing import LabelEncoder  # 标签编码器 / Кодировщик меток
from sklearn.model_selection import train_test_split  # 数据分割工具 / Инструмент разделения данных
from mlxtend.plotting import plot_decision_regions  # 决策边界可视化 / Визуализация границ решений
from gensim.models import Word2Vec  # 词向量模型 / Модель векторного представления слов

# ======================
# 2. 可视化函数定义 / Определение функций визуализации
# ======================
def plotScatterMatrix(df, plotSize, textSize):
    """
    绘制散点矩阵和相关系数
    Построение матрицы диаграмм рассеяния с коэффициентами корреляции
    """
    # 数据预处理步骤
    # Шаги предобработки данных
    df = df.select_dtypes(include=[np.number])  # 只保留数值型列 / Оставляем только числовые столбцы
    df = df.dropna(axis='columns')  # 删除含空值的列 / Удаляем столбцы с пропусками
    df = df[[col for col in df if df[col].nunique() > 1]]  # 保留有变化的列 / Оставляем столбцы с вариацией

    # 可视化逻辑
    # Логика визуализации
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal='hist')
    corrs = df.corr().values  # 计算相关系数矩阵 / Рассчитываем матрицу корреляций

    # 添加相关系数注释
    # Добавление аннотаций коэффициентов корреляции
    for i, j in zip(*plt.np.triu_indices_from(ax, k=1)):
        ax[i,j].annotate('%.1f'%corrs[i,j], (0.8,0.2),
                         xycoords='axes fraction',
                         ha='center', va='center',
                         size=textSize)
    plt.suptitle('Таблица анализа данных, коэффициент корреляции')
    plt.show()
```

```
# ==================
# 3. 数据加载与预处理 / Загрузка и предобработка данных
# ==================
# 加载原始数据 / Загрузка исходных данных
raw_table_data = pd.read_csv(
    'https://raw.githubusercontent.com/.../test_data_ECG.csv',
    nrows=5000  # 限制读取 5000 行 / Ограничение на чтение 5000 строк
)

# 异常值处理 / Обработка выбросов
columns_to_filter = ['rr_interval', 'p_onset', 'p_end', 'qrs_onset', 'qrs_end', 't_end', 'p_axis', 'qrs_axis', 't_axis']
full_df_filtered = raw_table_data[
    (raw_table_data[columns_to_filter] < 2000).all(axis=1)  # 过滤极端值 / Фильтрация экстремальных значений
]

# 文本报告处理 / Обработка текстовых отчетов
reports = [f'report_{x}' for x in range(18)]
full_df_filtered['report_0'] = full_df_filtered[reports].astype(str).agg(' '.join, axis=1)  # 合并文本列 / Объединение текстовых столбцов
full_df_filtered['report_0'] = full_df_filtered['report_0'].str.replace(r'\bnan\b', '', regex=True)  # 清理无效值 / Очистка невалидных значений

# ==================
# 4. 特征工程 / Инженерия признаков
# ==================
# Word2Vec 文本向量化 / Векторизация текста с Word2Vec
words = [text.split() for text in full_df_filtered['report']]
w2v_model = Word2Vec(
    words,
    vector_size=100,  # 向量维度 / Размерность векторов
    window=5,         # 上下文窗口 / Окно контекста
    min_count=1       # 最小词频 / Минимальная частота слова
)

def get_sentence_embedding(sentence):
    """生成句子向量 / Генерация векторного представления предложения"""
    word_vectors = [w2v_model.wv[word] for word in sentence.split() if word in w2v_model.wv]
    return np.mean(word_vectors, axis=0) if word_vectors else np.zeros(w2v_model.vector_size)

# 应用向量化 / Применение векторизации
full_df_filtered['report'] = full_df_filtered['report'].apply(lambda x: get_sentence_embedding(x).mean())

# ==================
# 5. 数据分析与可视化 / Анализ и визуализация данных
# ==================
# 箱线图分析 / Анализ ящичковыми диаграммами
```

```
plt.figure(figsize=(15,8))
sns.boxplot(data=table_data, x='Healthy_Status', y='rr_interval')
plt.title('Распределение RR-интервала по состоянию здоровья / RR Interval Distribution by Health Status')

# 主成分分析(PCA) / Анализ главных компонент
pca = PCA(n_components=2)
x_pca = pca.fit_transform(StandardScaler().fit_transform(table_data_pca))
plt.scatter(x_pca[:,0], x_pca[:,1], c=full_df_filtered['Healthy_Status'], cmap='viridis')

# ==================
# 6. 机器学习建模 / Построение ML-моделей
# ==================
# 数据分割 / Разделение данных
X_train, X_test, y_train, y_test = train_test_split(
    table_data.drop('Healthy_Status', axis=1),
    table_data['Healthy_Status'],
    test_size=0.3,
    stratify=table_data['Healthy_Status']  # 分层抽样 / Стратифицированная выборка
)

# 高斯朴素贝叶斯模型 / Модель наивного Байеса
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)

# 模型评估 / Оценка модели
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2%}")
print(classification_report(y_test, y_pred))

# ==================
# 7. AutoML 集成 / Интеграция AutoML
# ==================
# H2O AutoML 示例 / Пример с H2O AutoML
h2o.init()
automl = H2OAutoML(max_runtime_secs=60)
automl.train(x=x, y=y, training_frame=train)

# AutoGluon 示例 / Пример с AutoGluon
predictor = TabularPredictor(label='Healthy_Status').fit(train_data=auto_train_df)

# ==================
# 8. 高级分析 / Продвинутый анализ
# ==================
# t-SNE 可视化 / Визуализация t-SNE
tsne = TSNE(n_components=2, perplexity=30)
```

```
tsne_results = tsne.fit_transform(scaled_data)

sns.scatterplot(x=tsne_results[:,0], y=tsne_results[:,1], hue=full_df_filtered['Healthy_Status'])
```