

# **Asfendiyarov Kazakh National Medical University**

**Тема исследования: Практическое применение  
Автоматизированной системы научных исследований в медицине,  
здравоохранении и смежных областях**

**Проект: Анализ факторов риска сердечно сосудистых заболеваний  
и прогноз исходов лечения при помощи методов Машинного  
Обучения**

**Автор исследования: Dr. Alexander Wagner (Berlin)**





## Содержание

Предисловие .....	8
История создания Автоматизированной Системы Научных Исследований в медицине и здравоохранении «АСНИ-МЕД» .....	8
Краткая характеристика «АСНИ-МЕД» .....	8
Цель создание АСНИ .....	8
Назначение системы .....	9
Область применения системы.....	9
Задачи решаемые при помощи системы .....	9
Результаты применения системы .....	9
Введение.....	11
Краткое описание научного исследования .....	12
Цель .....	12
Метод.....	12
Результаты .....	12
Выводы.....	12
Введение в научное исследование.....	12
Материалы и методы .....	14
Этапы исследования.....	14
Исследовательский анализ данных (EDA) .....	14
Введение в EDA.....	14
Набор данных для анализа .....	17
Преобразование категориальной переменной в числовую .....	19
Результаты EDA .....	21
Ниже приведены результаты проведения ЕДА в форме таблиц и графиков. ....	21
Таблица №1. Исходные данные (выборка) для кардио-пациентов .....	21
Таблица №2. Описательная статистика читловых переменных набора данных для всех кардио-пациентов .....	22
График №1. BoxPlot для всех числовых переменных по классу Заболевание (HeartDisease).....	22
Таблица №3.....	23
График №2. BoxPlot для всех числовых переменных .....	25
Таблица №4.....	25
График №3. Распределение числовых переменных по классу пол (Sex) .....	27
Таблица №5.....	28
График №4. Гистограммы распределения для всех числовых переменных по классу Заболевание (HeartDisease).....	28
Таблица №6.....	29
График №5. Гистограммы распределения для всех числовых переменных по классу пол (Sex) .....	31



Таблица №7.....	31
График №6. Двумерное распределение переменной 'HeartDisease' по классам Sex и RestingBP.....	33
Таблица №8.....	33
График №7. График распределение переменной Cholesterol по Возрасту (Age).....	35
Таблица №9.....	35
График №8. Гистограммы распределения для всех числовых переменных, представленные на одном графике.....	37
Таблица №10.....	37
График №9. Матрица корреляции Пирсона для числовых переменных .....	39
Таблица №11 .....	39
График №10. Гистограммы распределения для всех числовых переменных, в виде субграфиков на одной панели .....	41
Таблица №12.....	42
График №11. Распределение переменной Cholesterol по классу (Age, Sex, FastingBS) в виде 4 субграфиков на одной панели .....	47
Таблица №13 .....	48
График №12. Распределение по возрасту (Age) для переменных: 'Sex','ChestPainType','FastingBS','RestingECG','ExerciseAngina','ST_Slope','HeartDisease' в форме Виалин-графиков .....	51
Таблица №14.....	52
График №13. Распределение всех числовых переменных в виде субграфиков по классу 'HeartDisease' на одной панели .....	56
График №14. Распределение всех числовых переменных в виде столбиковых диаграмм как субграфиков по классу 'HeartDisease' на одной панели .....	57
График №15. Распределение численности пациентов по всем переменным в виде столбиковых диаграмм как субграфиков по классу пол (Sex) на одной панели .....	58
График №16. Биполярное распределение переменной Cholesterol по возрасту 'Age' .....	59
График №17. Биполярное распределение разного графического типа всех численных переменных на одной панели.....	61
График №18. Плотность распределения переменной Cholesterol по классу пол (Sex) ....	63
График №19. Распределение переменной Age возрастным группам в виде столбиковых диаграм по классу 'HeartDisease' .....	65
График №20. Распределение переменной Cholesterol по возрастным группам в виде столбиковых диаграм по классу 'HeartDisease'.....	67
График №21. Распределение переменной RestingBP по возрастным группам в виде столбиковых диаграм по классу 'HeartDisease'.....	69
График №22. Распределение переменной MaxHR по возрастным группам в виде столбиковых диаграм по классу 'HeartDisease'.....	71
График №23. Распределение переменной Oldpeak по возрастным группам в виде столбиковых диаграм по классу 'HeartDisease'.....	73
График № 24. Торт-диаграмма распределения пациентов по класс-переменной HeartDisease.....	75
График №25. Торт-диаграмма распределения пациентов по класс-переменной Sex .....	77



График №26. Столбиковая диаграмма распределения переменных: 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope', 'HeartDisease' по категориям .....	79
График №27. Блок-бокс диаграмма распределения переменных: 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope', 'HeartDisease' в виде субграфиков на одной панели по категориям .....	81
График №28. Столбиковая диаграмма распределения переменных: 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope', 'HeartDisease' по категориям .....	83
График №29. Секторная диаграмма распределения переменной ChestPainType .....	85
График №30. Секторная диаграмма (2-го типа) распределения переменной ST_Slope ...	87
График №31. Комбинированная диаграмма распределения переменной пол (Sex) .....	89
График №32. Комбинированная диаграмма распределения переменной HeartDisease....	91
График №33. Столбиковая диаграмма распределения переменных: 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope' по категориям и классу 'HeartDisease'.....	93
График №34. Столбиковая диаграмма распределения переменных: 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope' по категориям и классу 'Sex' .....	95
График №35. Столбиковая диаграмма распределения переменных: 'Sex', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope', 'HeartDisease' по категориям и классу 'ChestPainType' .....	97
График №36. Комбинированная (столбиковая и секторная) диаграмма распределения переменной пол (Sex).....	99
График №37. Комбинированная (столбиковая и секторная) диаграмма распределения переменной ChestPainType .....	101
График №38. Комбинированная (столбиковая и секторная) диаграмма распределения переменной RestingECG .....	103
График №39. Комбинированная (столбиковая и секторная) диаграмма распределения переменной ExerciseAngina.....	105
График №40. Комбинированная (столбиковая и секторная) диаграмма распределения переменной ST_Slope.....	107
График №41. Комбинированная (столбиковая и секторная) диаграмма распределения переменной Cholesterol_Category .....	109
График №42. Комбинированная (столбиковая и секторная) диаграмма распределения переменной RestingBP_Category .....	111
График №43. Двойная секторная диаграмма (Sunburst, 6 субграфиков) распределения пар переменных: ['ChestPainType', 'FastingBS'], ['ST_Slope', 'RestingECG'], ['ExerciseAngina', 'ChestPainType'] по классу пол (Sex) .....	113
График №44. Столбиковая диаграмма распределения переменных: RestingECG, ChestPainType (2 субграфика на одной панели) по классу пол (Sex).....	115
График №45. Столбиковая диаграмма распределения переменных: ST_Slope, ExerciseAngina (2 субграфика на одной панели) по классу пол (Sex).....	117
Заключение по результатам EDA .....	119
Моделирование и Анализ данных .....	120



Результатами являются:.....	120
Модель: Linear Regression .....	120
Таблица №15. Таблица классификации .....	120
График №46. Confusion Matrix.....	121
График №47. ROC Curve .....	121
График №48. Score plot.....	122
Модель: Logistic Regression.....	122
Таблица №16. Таблица классификации .....	122
График №49. Confusion Matrix.....	123
График №50. ROC Curve .....	123
График №51. Score plot.....	124
Модель: Perceptron .....	124
Таблица №17. Таблица классификации .....	124
График №52. Confusion Matrix.....	125
График №53. ROC Curve .....	125
График №54. Score plot.....	126
Модель: Linear SVC .....	126
Таблица №18. Таблица классификации .....	126
График №55. Confusion Matrix.....	127
График №56. ROC Curve .....	127
График №57. Score plot.....	128
Модель: MLPClassifier .....	128
Таблица №19. Таблица классификации .....	128
График №58. Confusion Matrix.....	129
График №59. ROC Curve .....	129
График №60. Score plot.....	130
Модель: Decision Tree Classifier 1.....	130
Таблица №20. Таблица классификации .....	130
График №61. Confusion Matrix.....	131
График №62. ROC Curve .....	131
График №63. Score plot.....	132
Модель: Stochastic Gradient Decent.....	132
Таблица №21. Таблица классификации .....	132
График №64. Confusion Matrix.....	133
График №65. ROC Curve .....	133
График №66. Score plot.....	134
Модель: RidgeClassifier.....	134
Таблица №22. Таблица классификации .....	134
График №67. Confusion Matrix.....	135



График №68. ROC Curve .....	135
График №69. Score plot.....	136
Модель: BaggingClassifier.....	136
Таблица №23. Таблица классификации .....	136
График №70. Confusion Matrix.....	137
График №71. ROC Curve .....	137
График №72. Score plot.....	138
Модель: AdaBoostClassifier 1 .....	138
Таблица №24. Таблица классификации .....	138
График №73. Confusion Matrix.....	139
График №74. ROC Curve .....	139
График №75. Score plot.....	140
Модель: GradientBoostingClassifie .....	140
Таблица №25. Таблица классификации .....	140
График №76. Confusion Matrix.....	141
График №77. ROC Curve .....	141
График №78. Score plot.....	142
Модель: KNeighborsClassifier.....	142
Таблица №26. Таблица классификации .....	142
График №79. Confusion Matrix.....	143
График №80. ROC Curve .....	143
График №81. Score plot.....	144
Модель: DecisionTreeClassifier 2.....	144
Таблица №27. Таблица классификации .....	144
График №82. Confusion Matrix.....	145
График №83. ROC Curve .....	145
График №84. Score plot.....	146
Модель: RandomForestClassifier.....	146
Таблица №28. Таблица классификации .....	146
График №85. Confusion Matrix.....	147
График №86. ROC Curve .....	147
График №87. Score plot.....	148
Модель: XGBClassifier.....	148
Таблица №29. Таблица классификации .....	148
График №88. Confusion Matrix.....	149
График №89. ROC Curve .....	149
График №90. Score plot.....	150
Модель: AdaBoostClassifier 2 .....	150
Таблица №30. Таблица классификации .....	150



График №91. Confusion Matrix.....	151
График №92. ROC Curve .....	151
График №93. Score plot.....	152
Модель: Naive Bayes .....	152
Таблица №31. Таблица классификации .....	152
График №94. Confusion Matrix.....	153
График №95. ROC Curve .....	153
График №96. Score plot.....	154
Модель: SVC.....	154
Таблица №32. Таблица классификации .....	154
График №97. Confusion Matrix.....	155
График №98. ROC Curve .....	155
График №99. Score plot.....	156
Результаты моделирования .....	157
Оценка моделей и выбор наилучших для использования в диагности ССЗ .....	158
График №100. ROC-график для всех моделей .....	158
График №101 r2_score %. Линейный график А для всех моделей .....	158
График №102 ACC%. Линейный график В для всех моделей.....	158
График №103 rmse %. Линейный график С для всех моделей .....	159
Table №33. Характеристика лучших моделей после первого этапа.....	159
Table №34. Характеристика всех моделей после второго этапа.....	159
Table №35. Характеристика всех моделей после третьего этапа .....	159
График №104. График основных метрик для лучших моделей: AUC, F1, Precision, Accuracy_Test.....	161
График №105. График основных метрик для лучших моделей: Recall, r2_Train, Accuracy_Diff, RMSE_Train .....	162
Заключение .....	163
Рекомендации по применению АСНИ для исследователей.....	164
Актуальное состояние и дальнейшее развитие АСНИ .....	165
Литература .....	166
Дополнительные интернет источники .....	173
Приложение: Программный код.....	174
EDA Блок. Создание Таблиц и графиков.....	174
EDA Блок. Создание промежуточного отчета .....	193
Аналитический Блок. Проведение ML Анализа и сохранение выходных результатов.....	201
Заключительный системный Блок. Создание полного научного отчета .....	210



## Предисловие

Данная работа посвящена проблеме проведения научного исследования и создания научного отчета в медицине и здравоохранении при помощи Автоматизированной Системы Научных Исследований. Краткая характеристика системы приведена в данном разделе. Описание научного исследования, выполненное по установленным международным нормам, правилам и рекомендациям, приводится в последующих разделах данного документа.

## История создания Автоматизированной Системы Научных Исследований в медицине и здравоохранении «АСНИ-МЕД»

За многие десятилетия работы в роли Биостатистика и Data Scientist(a) в различных фирмах разной величины и масштаба и в разных странах у автора неоднократно возникала идея о необходимости автоматизировать процесс анализа данных. Со временем удалось реализовать несколько таких проектов и внедрить их в нескольких организациях. С примером одной из таких систем демонстрировавшейся на научной конференции (2019), можно познакомиться по ссылке:

[https://saswiki.de/display/KONFERENZEN/KSFE+2019?preview=19726371/19726410/23\\_KSFE\\_2019\\_Wagner\\_-](https://saswiki.de/display/KONFERENZEN/KSFE+2019?preview=19726371/19726410/23_KSFE_2019_Wagner_-)

[Ein SAS basiertes System zur automatisierten Auswertung und Berichterstellung von klinischen Studien.pdf](#)

С учетом накопленного опыта и в условиях появления новых возможностей в Информационных Технологиях, в том числе Web-технологий и Искусственного Интеллекта, возникли хорошие возможности для создания АСНИ собственными силами и в реальные сроки. Автор считает своим долгом предложить вашему вниманию концепт этой системы.

## Краткая характеристика «АСНИ-МЕД»

Научная работа – это наиболее сложная человеческая деятельность, для её успешного проведения требуются не только личные качества, но и соответствующий инструментарий. Прежде всего это научные данные и средства их анализа. В медико-социальных исследованиях этим инструментарием являются Биостатистика со всем набором современных методов анализа и прогноза данных и ситуаций и современные компьютерные системы, позволяющие это реализовать.

К сожалению, освоение этого технического инструментария требует больших затрат времени и не всем желающим это в силу различных причин удаётся. Эта проблема существует уже долгие годы и на протяжении длительного времени как крупные компании так и небольшие организации испытывают постоянную потребность в квалифицированных Биостатистиках, к сожалению, их поиск довольно трудное дело и не всегда завершается успехом. В то же время, привлечение специализированных организаций и Freelancer(ов) стоит дорого и не гарантируют от неудач.

## Цель создание АСНИ

Главная цель создание системы заключается в упрощении и облегчении процесса анализа медико-социальных данных и построение моделей анализа и прогноза ситуаций в процессах, происходящих в медицине, здравоохранении и в смежных областях (в медицинском страховании, в фармокондустрии и т.п.)

Цель реализуется через предоставление следующих возможностей пользователю:

- Простой и беспроблемный доступа к Базе статистических знаний
- Запуск типовых программ Анализа данных и прогноза ситуаций без необходимости инсталляции специальных статистических систем на пользовательском компьютере
- Самостоятельное решение своих (научных) проблем по Анализу медицинских данных и прогнозу ситуаций



- Предоставление других информационных и аналитических услуг

## Назначение системы

Система предназначена для широкого круга пользователя, в том числе:

- Студентов медицинских факультетов
- Аспирантов и докторантов медицинских факультетов
- Научных работников сферы здравоохранения
- Администраторов здравоохранения
- Сотрудников специализированных организаций, занимающихся анализом данных клинических исследований (Clinical Trials, <https://en.wikipedia.org/wiki/ClinicalTrials.gov>).
- Других пользователей

## Область применения системы

Данная Система предусмотрена для использования на всех уровнях обучения/науки и управления. Это означает, что как на государственном (с расширенным функциональным набором), так и на Вузовских, соответственно областных или территориальных уровнях система работает идентично. Привилегированный пользователь может использовать без ограничения все данные и возможности системы.

Другие пользователи имеют доступ только к данным конкретной области. Система построена по модульному принципу и является открытой для развития и расширения.

## Задачи решаемые при помощи системы

При помощи АСНИ решаются следующие задачи:

- Визуальный анализ данных при помощи интерактивных графиков и таблиц различного типа и свойства, отображающих в динамическом режиме например, состояние основных показателей здоровья населения территории
- Составление аналитических отчетов о состоянии параметров и показателей управляемого объекта, например, здоровья населения территории
- Научный анализ данных и прогноз при помощи современных средств продвинутой статистики (Advanced Statistics) и методов Искусственного Интеллекта
- Построения моделей оптимизации работы отрасли или организации
- Другие оперативные и стратегические задачи

### Методы решения задач АСНИ

Для решения поставленных задач АИС используются методы:

- Агрегации данных и создания многомерных отчетов
- Визуализации данных (динамические графики, карты, пр.)
- Классической прикладной математической статистики
- Методы статистического моделирования, в том числе Монте Карло
- Методы Доказательной медицины (Evidence Based Medicine)
- Методы Экономической медицины (Health Economics)
- Методы Искусственного Интеллекта, в том числе: Машинного обучения(Machine learning), глубокого обучения(Deep learning)
- Математические Методы оптимизации (Operations Research)
- Эвристические методы

## Результаты применения системы

Результатами являются:



## Автоматизированная Система Научных Исследований в медицине и здравоохранении «АСНИ-МЕД»

- Отчеты научно-исследовательских работ
- Статьи и презентации на научно-практических конференциях
- Диссертационные работы
- Отчеты плановых и коммерческих проектных работ
- Другие формы выходных результатов научных и проектных работ



## Введение

Цель данного научного проекта состоит из двух подцелей, в том числе:

- Первая главная подцель – это создание компьютерной системы АСНИ для автоматизированного анализа и прогноза ситуаций и феноменов из области медицины и здравоохранения при помощи современных информационных технологий и методов Машинного обучения, соответственно Искусственного Интеллекта.
- Вторая главная подцель – это непосредственное научное исследование определенной медицинской проблемы (Clinical Trials). В нашем случае это научное исследование оценки влияние различных факторов риска на заболеваемость сердечно-сосудистыми заболеваниями. Дальнейшее изложение посвящено этой второй подцели.



## Краткое описание научного исследования

### Цель

Для прогнозирования и диагностики основных неблагоприятных сердечно-сосудистых событий (major adverse cardiovascular events, MACE) используются стандартные методы Биостатистики, но их применение не всегда приводит к желаемому результату. В связи с этим в данной работе предлагается классификатор ансамбля мягкого голосования (SVE) с использованием алгоритмов машинного обучения (ML).

### Метод

Мы использовали набор данных [95] кардио-пациентов, а затем разделили его на обучающий (70%) и тестовый набор данных (30%). В-третьих, мы выбрали диапазоны гиперпараметров, чтобы найти наилучшую модель прогнозирования из 18 испытуемых. Мы сгенерировали каждую модель на основе машинного обучения с лучшими гиперпараметрами, а затем верифицировали с помощью тестового набора данных. Наконец, мы сравнили производительность в области под ROC-кривой (AUC), точность, прецизионность, полноту и F-оценку.

### Результаты

Окончательные результаты работы нашей системы представлены в соответствующих таблицах (№33-№35) и графиков (№100-№105), приведенных в подразделе «Оценка моделей и рекомендации»

### Выводы

Производительность нашей системы на базе моделей машинного обучения (RF, ET, GBM) была достаточно высокой, и ее основные прогностические факторы отличались высокой точностью. Данная работа будет использована нами в разработке и усовершенствования инструментария раннего прогнозирования и диагностики риска ССЗ у кардио-пациентов.

### Введение в научное исследование

За последние несколько десятилетий смертность пациентов с острым коронарным синдромом (ОКС) возросла [1] и стала ведущей причиной смертности во всем мире [2]. По данным Всемирной организации здравоохранения, острый коронарный синдром является основной причиной смерти во всем мире. Ранняя диагностика острого коронарного синдрома и его прогнозирование очень важны для пациентов с заболеваниями сердца. С другой стороны, очень трудно точно предсказать тяжесть острого коронарного синдрома по медицинскому набору данных, поскольку он зависит от множества факторов риска.

В 1960-х годах во Фрамингемском исследовании сердца [4] была выдвинута идея острого коронарного синдрома, и модель прогнозирования острого коронарного синдрома была разделена на два метода, а именно методы, основанные на регрессии, и методы, основанные на машинном обучении. Существует множество моделей прогнозирования риска, основанных на регрессиях, но наиболее распространенными моделями прогнозирования риска для раннего прогнозирования и диагностики основных неблагоприятных сердечно-сосудистых событий являются тромболизис при инфаркте миокарда [5] и Глобальный регистр острых коронарных событий [6], которые используются для прогнозирования оценки риска острого коронарного синдрома. Обе модели используют предыдущую медицинскую карту для изучения и прогнозирования тяжести состояния пациентов, но есть и некоторые недостатки этих старых моделей прогнозирования оценки риска, поскольку они были разработаны и внедрены около 10 лет назад. Эти модели используют несколько человек для прогнозирования риска и предсказывают уровень смертности на основе этих предикторов риска. Существует также больше предикторов, которые могут быть использованы для прогнозирования существования



серьезных неблагоприятных сердечно-сосудистых событий (**MACE**), таких как предыдущая медицинская карта и текущее состояние здоровья пациента.

Алгоритмы машинного обучения повышают точность прогнозирования сердечно-сосудистых заболеваний и предотвращают ненужное лечение [8]. Методы машинного обучения преодолели проблемы традиционных методов, основанных на регрессии, и популярны для диагностики и прогнозирования возникновения **MACE**. Кроме того, он устраняет типичные проблемы с данными и устраняет отсутствующие значения и выбросы с помощью методов интеллектуального анализа данных. Методы машинного обучения основаны на нелинейных связях и взаимодействиях между несколькими переменными и имеют дело с различными предикторами риска для точного прогнозирования риска пациентов. В этом исследовании также изучается эффективность методов прогнозирования риска на основе машинного обучения для прогнозирования степени тяжести пациентов с острым коронарным синдромом. Johnson et al. [9] отметили важность алгоритмов машинного обучения для прогнозирования и диагностики сердечно-сосудистых заболеваний. Тем не менее, методы, основанные на машинном обучении, имеют ряд сложных проблем для прогнозирования случаев **MACE**.

Во-первых, не существует специфического машинного обучения или ансамблевого подхода, что дает хорошие результаты для прогнозирования и работы с такого рода клиническими наборами данных. Кроме того, необходимо определить указанные предикторы, влияющие на возникновение острого коронарного синдрома и оказывающие большое влияние на **MACE**. К сожалению, старые модели прогнозирования в основном основаны на регрессии или их точность колеблется от 65 до 84% [10]. Кроме того, эти модели зависят от нескольких факторов риска. Существуют и другие факторы риска, которые оказывают большее влияние на возникновение острых коронарных синдромов. Кроме того, есть и другие факторы, которые мы должны вывести из других атрибутов и которые оказывают большое влияние на острый коронарный синдром.

Таким образом, в данной работе предлагается ансамблевый классификатор на основе машинного обучения с мягким голосованием, который может заниматься ранней диагностикой и прогнозом у пациентов с острым коронарным синдромом и обеспечить наилучший метод борьбы с возникновением сердечных событий. Основной целью данной работы является разработка модели прогнозирования риска для раннего выявления. Содержание нашего исследования также можно резюмировать следующим образом.

Во-первых, для экспериментов мы используем открытый набор данных [95]

Во-вторых, мы предлагаем классификатор ансамбля мягкого голосования с использованием 18 алгоритмов машинного обучения, с последующим отбором 7-ми лучших моделей для машинной диагностики и прогнозирования случаев:

1. Decision Tree Classifier 1
2. DecisionTreeClassifier 2
3. RandomForestClassifier
4. XGBClassifier
5. AdaBoostClassifier 2
6. BaggingClassifier
7. GradientBoostingClassifier

В-третьих, мы уточним предикторы риска **MACE** между применяемыми методами машинного обучения с помощью стандартных и общепринятых показателей эффективности моделей, таких как: точность, полнота, оценка F1 и площадь под ROC-кривой (AUC).



## Материалы и методы

### Этапы исследования

Наше исследование состоит из двух этапов.

#### Этап первый. Исследовательский анализ данных (EDA)

Исследовательский анализ включает в себя изучение данных и поиск связей между переменными, которые ранее были неизвестны. Вот что вам нужно знать:

- EDA помогает обнаруживать взаимосвязи между показателями в данных, которые не являются доказательством существования корреляции, как обозначается фразой «Корреляция не подразумевает причинно-следственную связь».
- Это полезно для обнаружения новых связей и формирования гипотез. Он управляет планированием проектирования и сбором данных.

Для предварительной обработки набора данных мы классифицировали все атрибутивные признаки по различным категориям, например, категориальные признаки, непрерывные признаки и дискретные признаки. Мы определили различные правила предварительной обработки для этих различных типов атрибутов. Для категориальных переменных мы применили кодирование меток [32], а также одно горячее кодирование [26] для предварительной обработки этих переменных. Для непрерывных атрибутов мы классифицировали набор данных по диапазонам, а затем применили кодировку меток для этих определенных подклассов. Для некоторых категориальных и непрерывных переменных, содержащих несколько значений, мы применили одно горячее кодирование, чтобы упростить управление значениями этих атрибутов. Одно горячее кодирование является одним из лучших решений для управления несколькими значениями и предварительной обработки тех атрибутов, которые содержат более одного параметра. В нашем наборе данных также есть атрибуты с двоичным значением. Для таких атрибутов, содержащих ровно два значения, мы преобразовали их в двоичную форму (0 и 1), обозначив 0 как No, 1 как Yes.

#### Этап второй. Моделирование и Анализ

Моделирование и Анализ заключается в проведение при помощи методов и алгоритмов (МЛ) исследования влияния факторов риска на заболеваемость пациентов ССЗ.

Ниже в последующих разделах приводятся детальное изложение обоих этапов исследования.

### Исследовательский анализ данных (EDA)

#### Введение в EDA

Исследовательский анализ данных (*Exploratory data analysis, EDA*) используется для анализа и исследования наборов данных и обобщения их основных характеристик.

EDA помогает определить, как лучше всего манипулировать входными данными для получения необходимых ответов, упрощая специалистам по обработке и анализу



данных обнаружение закономерностей, выявление аномалий, проверку гипотез или предположений.

EDA в основном используется для того, чтобы увидеть, что данные могут выявить за пределами формального моделирования или проверки гипотез, и обеспечивает лучшее понимание переменных набора данных и отношений между ними. Это также может помочь определить, подходят ли статистические методы, которые вы рассматриваете для анализа данных.

Основная цель EDA — помочь взглянуть на данные, прежде чем делать какие-либо предположения. Это может помочь выявить очевидные ошибки, а также лучше понять закономерности в данных, обнаружить выбросы или аномальные события, найти интересные связи между переменными.

Специалисты по обработке и анализу данных могут использовать исследовательский анализ, чтобы убедиться, что результаты, которые они получают, являются достоверными и применимыми к любым желаемым бизнес-результатам и целям. EDA также помогает заинтересованным сторонам, подтверждая, что они задают правильные вопросы. EDA может помочь ответить на вопросы о стандартных отклонениях, категориальных переменных и доверительных интервалах.

Конкретные статистические функции и методы, которые можно выполнять с помощью инструментов EDA, включают:

- Методы кластеризации и уменьшения размерности, которые помогают создавать графические представления многомерных данных, содержащих множество переменных.
- Одномерная визуализация каждого поля в необработанном наборе данных со сводной статистикой.
- Двумерные визуализации и сводная статистика, которые позволяют оценить связь между каждой переменной в наборе данных и целевой переменной, которую вы просматриваете.
- Многомерные визуализации для сопоставления и понимания взаимодействий между различными полями данных.
- Кластеризация K-средних — это метод кластеризации в , при котором точки данных распределяются по K-группам, т.е. по количеству кластеров, в зависимости от расстояния от центроида каждой группы. Точки данных, ближайшие к определенному центроиду, будут объединены в одну категорию. Кластеризация K-средних обычно используется для сегментации рынка, распознавания образов и сжатия изображений.
- Прогностические модели, такие как линейная регрессия, используют статистику и данные для прогнозирования результатов.

Существует четыре основных типа EDA:

**Одномерный неграфический.** Это простейшая форма анализа данных, при которой анализируемые данные состоят всего из одной переменной. Поскольку это одна переменная, она не имеет отношения к причинам или отношениям. Основной целью одномерного анализа является описание данных и поиск закономерностей, которые в них существуют.



**Одномерная графика.** Неграфические методы не дают полной картины данных. Поэтому требуются графические методы. К распространенным типам одномерной графики относятся:

Штамбовые и листовые графики, которые показывают все значения данных и форму распределения.

Гистограммы, гистограмма, на которой каждый столбец представляет частоту (количество) или долю (количество/общее количество) вариантов для диапазона значений.

Ящичковые диаграммы, которые графически изображают сводку из пяти чисел минимума, первого квартиля, медианы, третьего квартиля и максимума.

**Многомерные неграфические:** Многомерные данные возникают из нескольких переменных. Многомерные неграфические методы САПР обычно показывают взаимосвязь между двумя или более переменными данных с помощью перекрестной таблицы или статистики.

**Многовариантная графика:** Многомерные данные используют графику для отображения связей между двумя или более наборами данных. Наиболее часто используемым рисунком является сгруппированная линейчатая диаграмма или линейчатая диаграмма, где каждая группа представляет один уровень одной из переменных, а каждая полоса в группе представляет уровни другой переменной.

К другим распространенным типам многомерной графики относятся:

Точечная диаграмма, которая используется для отображения точек данных по горизонтальной и вертикальной оси, чтобы показать, насколько одна переменная подвержена влиянию другой.

Многомерная диаграмма, представляющая собой графическое представление взаимосвязей между факторами и реакцией.

Запустите диаграмму, которая представляет собой линейный график данных, построенный во времени.

Пузырьковая диаграмма, представляющая собой визуализацию данных, отображающую несколько кругов (пузырьков) на двумерном графике.

Тепловая карта, представляющая собой графическое представление данных, где значения изображены цветом.



## Набор данных для анализа

Краткое описание переменных набора входных данных[95]

**Age:** age of the patient [years]

**Sex:** sex of the patient [M: Male, F: Female]

**ChestPainType:** chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]

**RestingBP:** resting blood pressure [mm Hg]

**Cholesterol:** serum cholesterol [mm/dl]

**FastingBS:** fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]

**RestingECG:** resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]

**MaxHR:** maximum heart rate achieved [Numeric value between 60 and 202]

**ExerciseAngina:** exercise-induced angina [Y: Yes, N: No]

**Oldpeak:** oldpeak = ST [Numeric value measured in depression]

**ST\_Slope:** the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]

**HeartDisease:** output class [1: heart disease, 0: Normal]

Наш набор данных содержит в общей сложности 6 числовых переменных:

**Age, RestingBP, Cholesterol, MaxHR, Oldpeak, HeartDisease**

В нашем наборе данных имеется также 6 категориальных переменных:

**Sex, ChestPainType, FastingBS, RestingECG, ExerciseAngina, ST\_Slope**





## Преобразование категориальной переменной в числовую

Категориальные значения — это тип данных, которые могут быть сгруппированы в различные категории, такие как пол, цвет или жанр. Числовые значения — это данные, которые имеют числовое значение, например возраст, рост или цена. Нам необходимо преобразовать категориальные значения в числовые в САПР по нескольким причинам:

Требования к моделированию: При планировании создания прогнозных моделей или моделей машинного обучения (МО), необходимы числовые представления. Многие алгоритмы машинного обучения, особенно основанные на математических уравнениях, работают с числовыми входными данными.

Визуализация: Числовые данные часто легче визуализировать и интерпретировать. Графики, диаграммы и другие визуализации обычно используются для получения аналитических сведений о данных, а числовые представления облегчают создание осмысленных визуализаций данных.

Статистический анализ: Некоторые статистические тесты и анализы предполагают наличие числовых данных. Например, коэффициенты корреляции, регрессионный анализ и другие статистические методы предназначены для работы с числовыми переменными.

Согласованность типов данных: преобразование категориальных переменных в числовые представления помогает поддерживать согласованность типов данных в наборе данных. Такая согласованность может упростить процесс анализа данных и сделать его более понятным.

Совместимость с методами анализа: Многие статистические методы и методы машинного обучения требуют числовых входных данных. Преобразование категориальных переменных в числовые представления позволяет применять к набору данных более широкий спектр методов анализа.

Существует два распространенных метода преобразования категориальных переменных в числовые:

One-Hot Encoding: Этот метод создает двоичные столбцы для каждой категории, представляющие наличие или отсутствие этой категории.

Кодировка метки: Этот метод присваивает каждой категории уникальное целое число.

Однако горячее кодирование более распространено для двоичных категориальных переменных, так как оно явно представляет каждую категорию независимо.

Здесь мы используем метод One-Hot Encoding для преобразования данных.





## Результаты ЕДА

Ниже приведены результаты проведения ЕДА в форме таблиц и графиков.

**Таблица №1. Исходные данные (выборка) для кардио-пациентов**

Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
M	ATA	140	289	0	Normal	172	N	0	Up	0
F	NAP	160	180	0	Normal	156	N	1	Flat	1
M	ATA	130	283	0	ST	98	N	0	Up	0
F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
M	NAP	150	195	0	Normal	122	N	0	Up	0
M	NAP	120	339	0	Normal	170	N	0	Up	0
F	ATA	130	237	0	Normal	170	N	0	Up	0
M	ATA	110	208	0	Normal	142	N	0	Up	0
M	ASY	140	207	0	Normal	130	Y	1.5	Flat	1
F	ATA	120	284	0	Normal	120	N	0	Up	0
F	NAP	130	211	0	Normal	142	N	0	Up	0
M	ATA	136	164	0	ST	99	Y	2	Flat	1
M	ATA	120	204	0	Normal	145	N	0	Up	0
M	ASY	140	234	0	Normal	140	Y	1	Flat	1
F	NAP	115	211	0	ST	137	N	0	Up	0
F	ATA	120	273	0	Normal	150	N	1.5	Flat	0
M	ASY	110	196	0	Normal	166	N	0	Flat	1
F	ATA	120	201	0	Normal	165	N	0	Up	0
M	ASY	100	248	0	Normal	125	N	1	Flat	1
M	ATA	120	267	0	Normal	160	N	3	Flat	1
F	TA	100	223	0	Normal	142	N	0	Up	0
M	ATA	120	184	0	Normal	142	N	1	Flat	0
F	ATA	124	201	0	Normal	164	N	0	Up	0
M	ATA	150	288	0	Normal	150	Y	3	Flat	1
M	NAP	130	215	0	Normal	138	N	0	Up	0
M	NAP	130	209	0	Normal	178	N	0	Up	0
M	ASY	124	260	0	ST	112	Y	3	Flat	0
M	ATA	120	284	0	Normal	118	N	0	Up	0
F	ATA	113	468	0	Normal	127	N	0	Up	0



**Таблица №2. Описательная статистика числовых переменных набора данных для всех кардио- пациентов**

Variable	count	mean	std	min	25%	50%	75%	max
Age	918	53.51	9.43	28	47	54	60	77
RestingBP	918	132.4	18.51	0	120	130	140	200
Cholesterol	918	198.8	109.38	0	173.25	223	267	603
FastingBS	918	0.23	0.42	0	0	0	0	1
MaxHR	918	136.81	25.46	60	120	138	156	202
Oldpeak	918	0.89	1.07	-2.6	0	0.6	1.5	6.2
HeartDisease	918	0.55	0.5	0	0	1	1	1

**График №1. BoxPlot для всех числовых переменных по классу Заболевание (HeartDisease)**

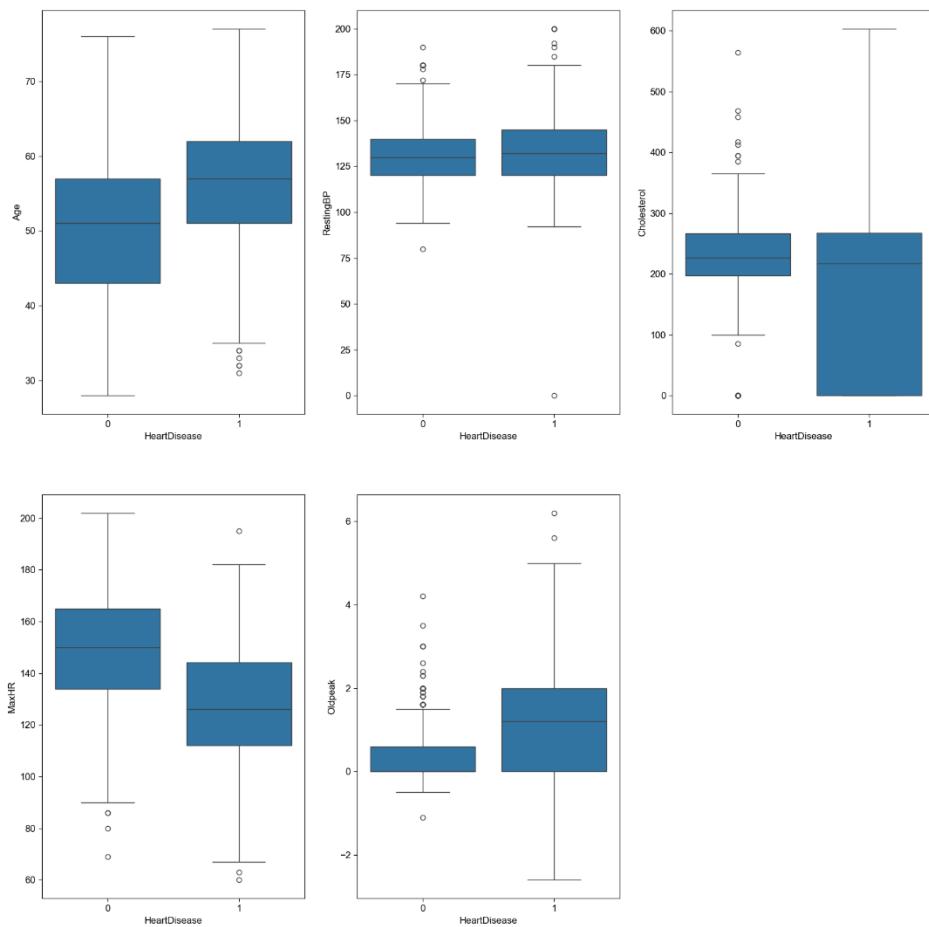




Таблица №3

Analysis: Cholesterol by Geschlecht, Alter								
Female	0 - 29 years	4	132	263	131	211	224	58
	30 - <39 years	49	117	529	412	243	240	68
	40 - <65 years	448	85	603	518	241	233	57
	65 - <75 years	57	110	384	274	243	245	51
	75+ years	6	113	310	197	221	214	77
Male	30 - <39 years	17	160	275	115	207	210	34
	40 - <65 years	148	141	468	327	259	253	58
	65 - <75 years	16	149	564	415	281	267	97
	75+ years	1	197	197	0	197	197	
Total	0 - 29 years	4	132	263	131	211	224	58
	30 - <39 years	66	117	529	412	234	220	63
	40 - <65 years	596	85	603	518	246	238	58
	65 - <75 years	73	110	564	454	251	246	66
	75+ years	7	113	310	197	218	203	70

© Dr. Alexander Wagner. Все права охраняются законом





## График №2. BoxPlot для всех числовых переменных Boxplots for each variable

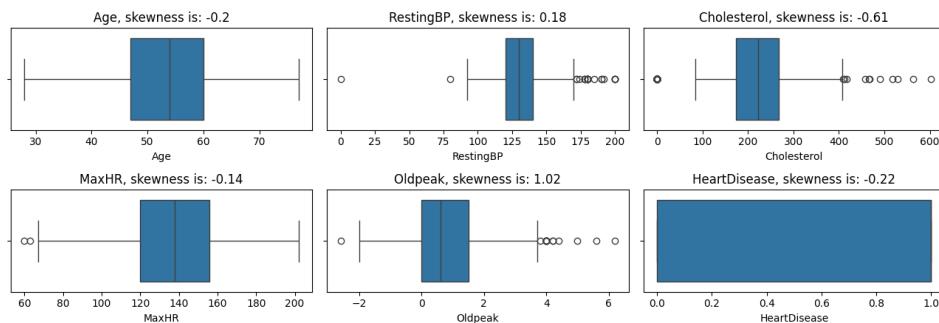


Таблица №4

Analysis: RestingBP by Geschlecht, Alter								
		N	Min	Max	Range	Mean	Median	Std
<b>Female</b>	<b>0 - 29 years</b>	4	120	140	20	130	130	8
	<b>30 - &lt;39 years</b>	57	92	190	98	124	120	17
	<b>40 - &lt;65 years</b>	579	80	200	120	132	130	18
	<b>65 - &lt;75 years</b>	78	100	180	80	140	140	16
	<b>75+ years</b>	6	104	170	66	137	131	25
<b>Male</b>	<b>30 - &lt;39 years</b>	19	94	170	76	121	120	18
	<b>40 - &lt;65 years</b>	155	95	200	105	133	130	19
	<b>65 - &lt;75 years</b>	18	106	178	72	141	148	21
	<b>75+ years</b>	1	140	140	0	140	140	
<b>Total</b>	<b>0 - 29 years</b>	4	120	140	20	130	130	8
	<b>30 - &lt;39 years</b>	76	92	190	98	124	120	17
	<b>40 - &lt;65 years</b>	734	80	200	120	132	130	18
	<b>65 - &lt;75 years</b>	96	100	180	80	140	140	17
	<b>75+ years</b>	7	104	170	66	137	136	22

© Dr. Alexander Wagner. Все права охраняются законом





### График №3. Распределение числовых переменных по классу пол (Sex)

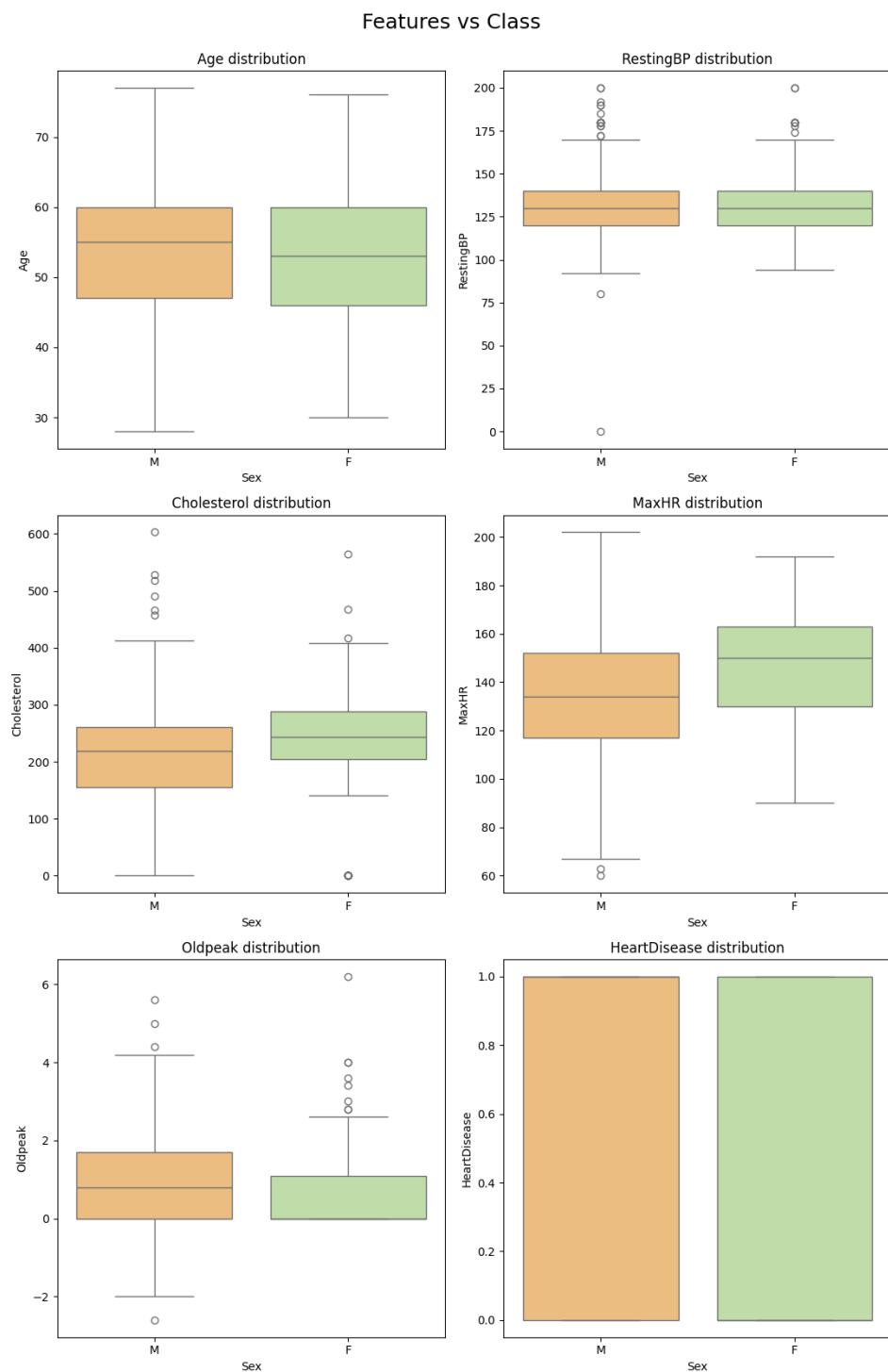


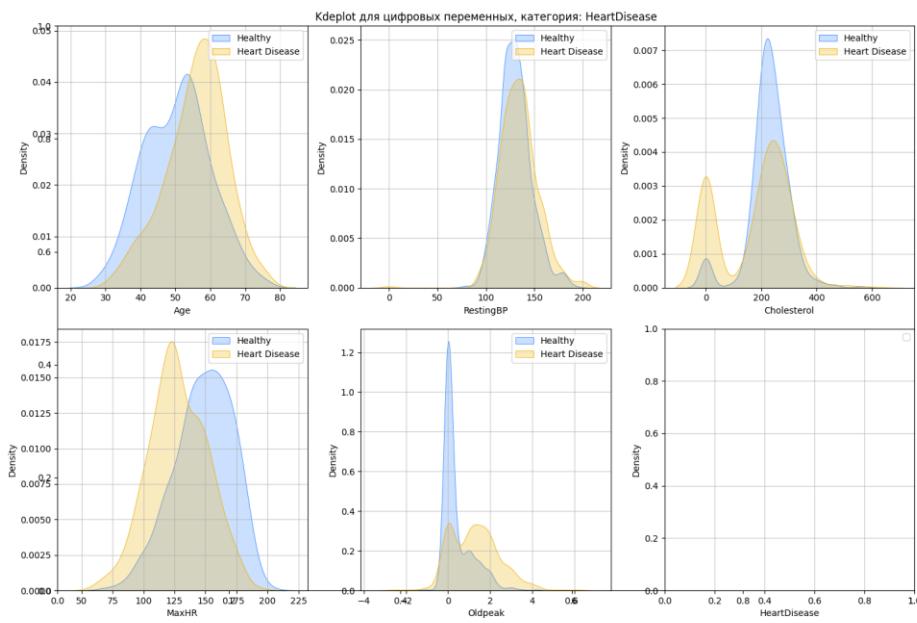


Таблица №5

		Analysis: MaxHR by Geschlecht, Alter							
		N	Min	Max	Range	Mean	Median	Std	
Female	0 - 29 years	4	160	202	42	179	178	18	
	30 - <39 years	57	98	187	89	153	154	23	
	40 - <65 years	580	60	195	135	134	134	25	
	65 - <75 years	78	67	174	107	123	124	21	
	75+ years	6	108	162	54	121	112	21	
Male	30 - <39 years	19	129	192	63	164	166	19	
	40 - <65 years	155	90	180	90	145	148	21	
	65 - <75 years	18	90	172	82	138	145	23	
	75+ years	1	116	116	0	116	116		
Total	0 - 29 years	4	160	202	42	179	178	18	
	30 - <39 years	76	98	192	94	156	156	22	
	40 - <65 years	735	60	195	135	136	138	25	
	65 - <75 years	96	67	174	107	126	125	22	
	75+ years	7	108	162	54	120	112	19	

© Dr. Alexander Wagner. Все права охраняются законом

График №4. Гистограммы распределения для всех числовых переменных по классу Заболевание (HeartDisease)





## Таблица №6

<i>Analysis: Oldpeak by Geschlecht, Alter</i>		N	Min	Max	Range	Mean	Median	Std
Female	<b>30 - &lt;39 years</b>	19	0	4	4	2	1	1
	<b>40 - &lt;65 years</b>	356	0	6	6	2	2	1
	<b>65 - &lt;75 years</b>	63	0	4	4	2	2	1
	<b>75+ years</b>	4	2	4	2	3	3	1
Male	<b>30 - &lt;39 years</b>	4	1	3	2	1	1	1
	<b>40 - &lt;65 years</b>	77	0	6	6	1	1	1
	<b>65 - &lt;75 years</b>	13	0	3	2	1	1	1
	<b>75+ years</b>	1	1	1	0	1	1	
Total	<b>30 - &lt;39 years</b>	23	0	4	4	2	1	1
	<b>40 - &lt;65 years</b>	433	0	6	6	2	2	1
	<b>65 - &lt;75 years</b>	76	0	4	4	1	2	1
	<b>75+ years</b>	5	1	4	2	2	2	1

© Dr. Alexander Wagner. Все права охраняются законом





## График №5. Гистограммы распределения для всех числовых переменных по классу пол (Sex)

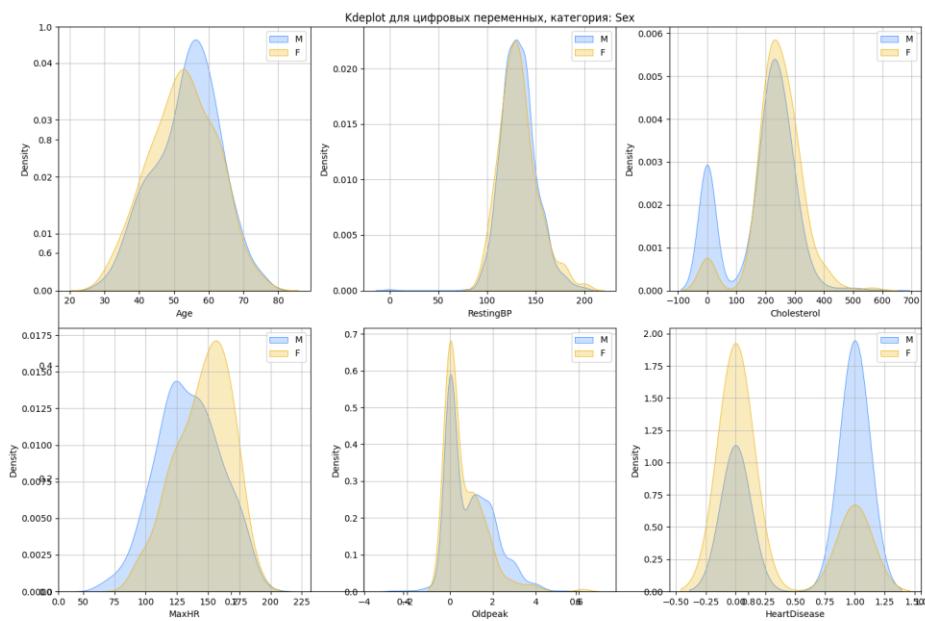


Таблица №7

Age	Healthy			Heart Disease			Всего			
	Mean	N	%	Mean	N	%	Mean	N	%	
Female	0 - 29 years	28.75	4	100.00			0.00	28.75	4	100.00
	30 - <39 years	36.68	34	59.65	36.17	23	40.35	36.47	57	100.00
	40 - <65 years	51.41	213	36.72	54.73	367	63.28	53.51	580	100.00
	65 - <75 years	67.80	15	19.23	68.10	63	80.77	68.04	78	100.00
	75+ years	75.00	1	16.67	76.00	5	83.33	75.83	6	100.00
	Всего	50.20	267	36.83	55.87	458	63.17	53.78	725	100.00
Male	0 - 29 years									
	30 - <39 years	35.56	16	84.21	36.33	3	15.79	35.68	19	100.00
	40 - <65 years	51.12	112	72.26	56.51	43	27.74	52.61	155	100.00
	65 - <75 years	68.00	14	77.78	67.50	4	22.22	67.89	18	100.00
	75+ years	76.00	1	100.00			0.00	76.00	1	100.00
	Всего	51.20	143	74.09	56.18	50	25.91	52.49	193	100.00
Total	0 - 29 years	28.75	4	100.00			0.00	28.75	4	100.00
	30 - <39 years	36.32	50	65.79	36.19	26	34.21	36.28	76	100.00
	40 - <65 years	51.31	325	44.22	54.92	410	55.78	53.32	735	100.00
	65 - <75 years	67.90	29	30.21	68.06	67	69.79	68.01	96	100.00
	75+ years	75.50	2	28.57	76.00	5	71.43	75.86	7	100.00
	Всего	50.55	410	44.66	55.90	508	55.34	53.51	918	100.00



Age		Healthy			Heart Disease			Всего		
		Mean	N	%	Mean	N	%	Mean	N	%
Всего	0 - 29 years	28.75	8	100.00			0.00	28.75	8	100.00
	30 - <39 years	36.32	100	65.79	36.19	52	34.21	36.28	152	100.00
	40 - <65 years	51.31	650	44.22	54.92	820	55.78	53.32	1470	100.00
	65 - <75 years	67.90	58	30.21	68.06	134	69.79	68.01	192	100.00
	75+ years	75.50	4	28.57	76.00	10	71.43	75.86	14	100.00
	Всего	50.55	820	44.66	55.90	1016	55.34	53.51	1836	100.00

© Dr. Alexander Wagner. Все права охраняются законом



## График №6. Двумерное распределение переменной 'HeartDisease' по классам Sex и RestingBP

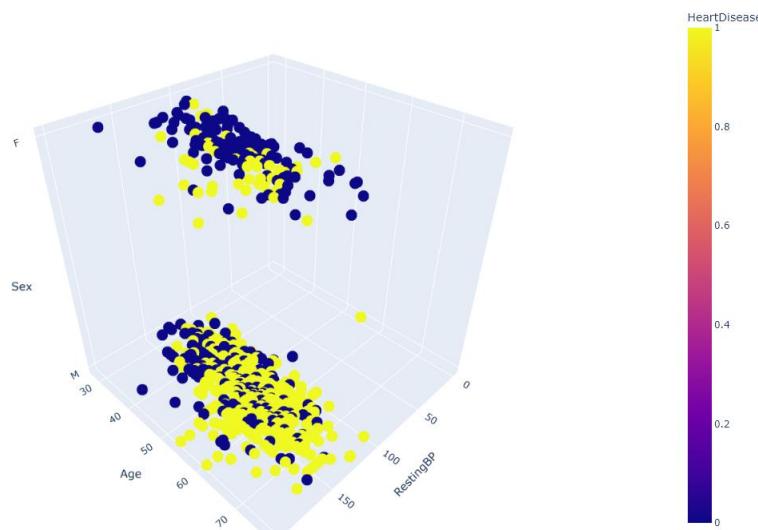


Таблица №8

Cholesterol		Healthy			Heart Disease			Всего			
		Mean	N	%	Mean	N	%	Mean	N	%	
Female	0 - 29 years	210.50	4	100.00				0.00	210.50	4	100.00
	30 - <39 years	240.91	33	67.35	248.13	16	32.65	243.27	49	100.00	
	40 - <65 years	231.16	199	44.42	249.12	249	55.58	241.14	448	100.00	
	65 - <75 years	239.45	11	19.30	243.33	46	80.70	242.58	57	100.00	
	75+ years	310.00	1	16.67	203.20	5	83.33	221.00	6	100.00	
	Всего	232.81	248	43.97	247.50	316	56.03	241.04	564	100.00	
Male	0 - 29 years										
	30 - <39 years	204.38	16	94.12	246.00	1	5.88	206.82	17	100.00	
	40 - <65 years	251.06	111	75.00	282.92	37	25.00	259.03	148	100.00	
	65 - <75 years	289.21	14	87.50	226.50	2	12.50	281.38	16	100.00	
	75+ years	197.00	1	100.00			0.00	197.00	1	100.00	
	Всего	249.18	142	78.02	279.18	40	21.98	255.77	182	100.00	
Total	0 - 29 years	210.50	4	100.00				0.00	210.50	4	100.00
	30 - <39 years	228.98	49	74.24	248.00	17	25.76	233.88	66	100.00	
	40 - <65 years	238.28	310	52.01	253.50	286	47.99	245.58	596	100.00	
	65 - <75 years	267.32	25	34.25	242.63	48	65.75	251.08	73	100.00	
	75+ years	253.50	2	28.57	203.20	5	71.43	217.57	7	100.00	
	Всего	238.77	390	52.28	251.06	356	47.72	244.64	746	100.00	



Cholesterol		Healthy			Heart Disease			Всего		
		Mean	N	%	Mean	N	%	Mean	N	%
<b>Всего</b>	<b>0 - 29 years</b>	210.50	8	100.00			0.00	210.50	8	100.00
	<b>30 - &lt;39 years</b>	228.98	98	74.24	248.00	34	25.76	233.88	132	100.00
	<b>40 - &lt;65 years</b>	238.28	620	52.01	253.50	572	47.99	245.58	1192	100.00
	<b>65 - &lt;75 years</b>	267.32	50	34.25	242.63	96	65.75	251.08	146	100.00
	<b>75+ years</b>	253.50	4	28.57	203.20	10	71.43	217.57	14	100.00
	<b>Всего</b>	238.77	780	52.28	251.06	712	47.72	244.64	1492	100.00

© Dr. Alexander Wagner. Все права охраняются законом



## График №7. График распределение переменной Cholesterol по Возрасту (Age)

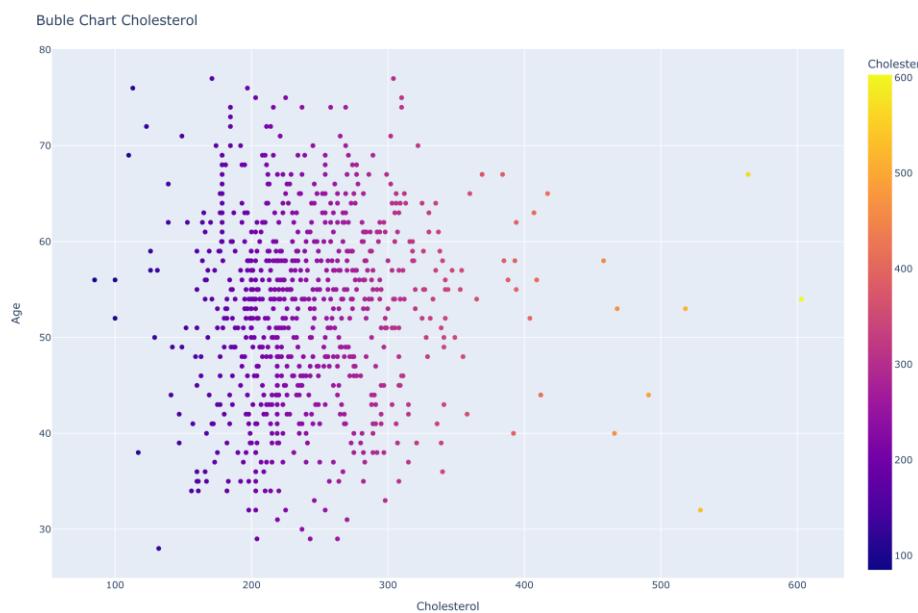


Таблица №9

RestingBP		Healthy			Heart Disease			Всего			
		Mean	N	%	Mean	N	%	Mean	N	%	
Female	0 - 29 years	130.00	4	100.00				0.00	130.00	4	100.00
	30 - <39 years	128.50	34	59.65	118.43	23	40.35	124.44	57	100.00	
	40 - <65 years	130.69	213	36.79	133.52	366	63.21	132.47	579	100.00	
	65 - <75 years	138.13	15	19.23	139.94	63	80.77	139.59	78	100.00	
	75+ years	160.00	1	16.67	131.80	5	83.33	136.50	6	100.00	
	Всего	130.93	267	36.88	133.62	457	63.12	132.63	724	100.00	
Male	0 - 29 years										
	30 - <39 years	123.94	16	84.21	105.00	3	15.79	120.95	19	100.00	
	40 - <65 years	128.64	112	72.26	142.84	43	27.74	132.58	155	100.00	
	65 - <75 years	134.71	14	77.78	160.75	4	22.22	140.50	18	100.00	
	75+ years	140.00	1	100.00			0.00	140.00	1	100.00	
	Всего	128.79	143	74.09	142.00	50	25.91	132.21	193	100.00	
Total	0 - 29 years	130.00	4	100.00				0.00	130.00	4	100.00
	30 - <39 years	127.04	50	65.79	116.88	26	34.21	123.57	76	100.00	
	40 - <65 years	129.98	325	44.28	134.50	409	55.72	132.50	734	100.00	
	65 - <75 years	136.48	29	30.21	141.18	67	69.79	139.76	96	100.00	
	75+ years	150.00	2	28.57	131.80	5	71.43	137.00	7	100.00	
	Всего	130.18	410	44.71	134.45	507	55.29	132.54	917	100.00	



RestingBP		Healthy			Heart Disease			Всего		
		Mean	N	%	Mean	N	%	Mean	N	%
<b>Всего</b>	<b>0 - 29 years</b>	130.00	8	100.00			0.00	130.00	8	100.00
	<b>30 - &lt;39 years</b>	127.04	100	65.79	116.88	52	34.21	123.57	152	100.00
	<b>40 - &lt;65 years</b>	129.98	650	44.28	134.50	818	55.72	132.50	1468	100.00
	<b>65 - &lt;75 years</b>	136.48	58	30.21	141.18	134	69.79	139.76	192	100.00
	<b>75+ years</b>	150.00	4	28.57	131.80	10	71.43	137.00	14	100.00
	<b>Всего</b>	130.18	820	44.71	134.45	1014	55.29	132.54	1834	100.00

© Dr. Alexander Wagner. Все права охраняются законом



## График №8. Гистограммы распределения для всех числовых переменных, представленные на одном графике

EDA на наборе данных показала нам, как каждая переменная связана с переменной отклика и как мы можем сделать нашу модель эффективной, используя различные методы EDA. Визуализации данных поднимают наше понимание набора данных на более высокий уровень, позволяя нам делать выводы.

Интеграция классификатора дерева принятия решений в наш анализ расширяет прогностические возможности нашей модели. В этом блоге мы не только изучили набор данных с помощью методов EDA, но и сделали еще один шаг вперед, внедрив модель машинного обучения. Такой целостный подход позволяет нам использовать сильные стороны как статистического анализа, так и прогнозного моделирования, способствуя более глубокому пониманию сложной динамики, связанной с экстремальными погодными явлениями.

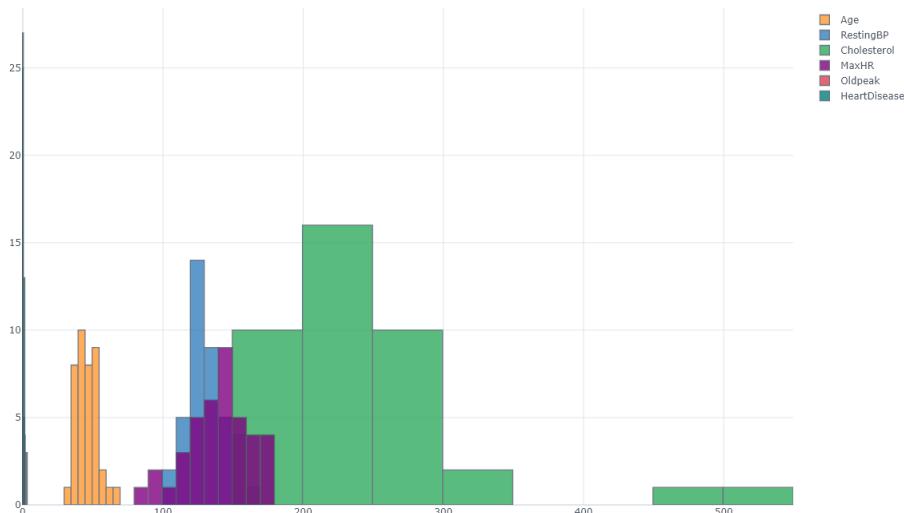


Таблица №10

MaxHR		Healthy			Heart Disease			Всего		
		Mean	N	%	Mean	N	%	Mean	N	%
Female	0 - 29 years	179.25	4	100.00			0.00	179.25	4	100.00
	30 - <39 years	159.53	34	59.65	143.91	23	40.35	153.23	57	100.00
	40 - <65 years	146.34	213	36.72	126.53	367	63.28	133.80	580	100.00
	65 - <75 years	133.60	15	19.23	120.65	63	80.77	123.14	78	100.00
	75+ years	112.00	1	16.67	122.40	5	83.33	120.67	6	100.00
	Всего	147.67	267	36.83	126.55	458	63.17	134.33	725	100.00



MaxHR		Healthy			Heart Disease			Всего		
		Mean	N	%	Mean	N	%	Mean	N	%
Male	<b>0 - 29 years</b>									
	<b>30 - &lt;39 years</b>	165.63	16	84.21	157.33	3	15.79	164.32	19	100.00
	<b>40 - &lt;65 years</b>	147.75	112	72.26	137.88	43	27.74	145.01	155	100.00
	<b>65 - &lt;75 years</b>	142.86	14	77.78	122.50	4	22.22	138.33	18	100.00
	<b>75+ years</b>	116.00	1	100.00			0.00	116.00	1	100.00
	<b>Всего</b>	149.05	143	74.09	137.82	50	25.91	146.14	193	100.00
Total	<b>0 - 29 years</b>	179.25	4	100.00			0.00	179.25	4	100.00
	<b>30 - &lt;39 years</b>	161.48	50	65.79	145.46	26	34.21	156.00	76	100.00
	<b>40 - &lt;65 years</b>	146.83	325	44.22	127.72	410	55.78	136.17	735	100.00
	<b>65 - &lt;75 years</b>	138.07	29	30.21	120.76	67	69.79	125.99	96	100.00
	<b>75+ years</b>	114.00	2	28.57	122.40	5	71.43	120.00	7	100.00
	<b>Всего</b>	148.15	410	44.66	127.66	508	55.34	136.81	918	100.00
Всего	<b>0 - 29 years</b>	179.25	8	100.00			0.00	179.25	8	100.00
	<b>30 - &lt;39 years</b>	161.48	100	65.79	145.46	52	34.21	156.00	152	100.00
	<b>40 - &lt;65 years</b>	146.83	650	44.22	127.72	820	55.78	136.17	1470	100.00
	<b>65 - &lt;75 years</b>	138.07	58	30.21	120.76	134	69.79	125.99	192	100.00
	<b>75+ years</b>	114.00	4	28.57	122.40	10	71.43	120.00	14	100.00
	<b>Всего</b>	148.15	820	44.66	127.66	1016	55.34	136.81	1836	100.00

© Dr. Alexander Wagner. Все права охраняются законом



## График №9. Матрица корреляции Пирсона для числовых переменных

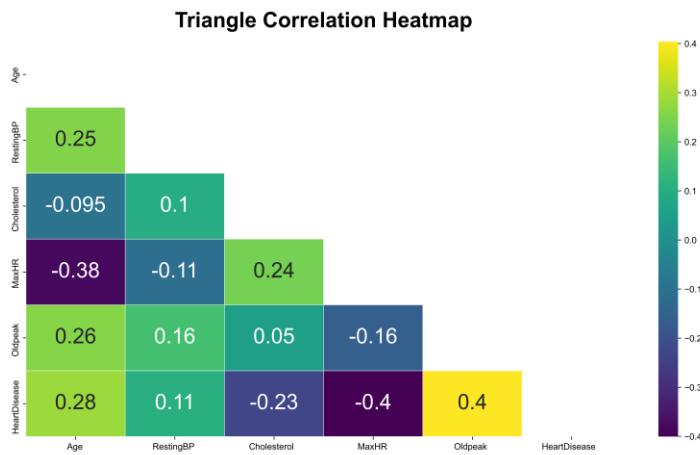


Таблица №11

Oldpeak		Healthy			Heart Disease			Всего		
		Mean	N	%	Mean	N	%	Mean	N	%
Female	30 - <39 years	1.70	5	26.32	1.53	14	73.68	1.57	19	100.00
	40 - <65 years	1.04	84	23.60	1.73	272	76.40	1.57	356	100.00
	65 - <75 years	0.76	12	19.05	1.76	51	80.95	1.57	63	100.00
	75+ years	2.00	1	25.00	2.83	3	75.00	2.63	4	100.00
	Всего	1.05	102	23.08	1.74	340	76.92	1.58	442	100.00
Male	30 - <39 years	1.05	2	50.00	1.90	2	50.00	1.47	4	100.00
	40 - <65 years	1.02	46	59.74	1.97	31	40.26	1.40	77	100.00
	65 - <75 years	1.13	11	84.62	1.00	2	15.38	1.11	13	100.00
	75+ years	1.10	1	100.00			0.00	1.10	1	100.00
	Всего	1.04	60	63.16	1.91	35	36.84	1.36	95	100.00
Total	30 - <39 years	1.51	7	30.43	1.58	16	69.57	1.56	23	100.00
	40 - <65 years	1.03	130	30.02	1.76	303	69.98	1.54	433	100.00
	65 - <75 years	0.93	23	30.26	1.73	53	69.74	1.49	76	100.00
	75+ years	1.55	2	40.00	2.83	3	60.00	2.32	5	100.00
	Всего	1.04	162	30.17	1.76	375	69.83	1.54	537	100.00
Всего	30 - <39 years	1.51	14	30.43	1.58	32	69.57	1.56	46	100.00
	40 - <65 years	1.03	260	30.02	1.76	606	69.98	1.54	866	100.00
	65 - <75 years	0.93	46	30.26	1.73	106	69.74	1.49	152	100.00
	75+ years	1.55	4	40.00	2.83	6	60.00	2.32	10	100.00
	Всего	1.04	324	30.17	1.76	750	69.83	1.54	1074	100.00

© Dr. Alexander Wagner. Все права охраняются законом





## График №10. Гистограммы распределения для всех числовых переменных, в виде субграфиков на одной панели

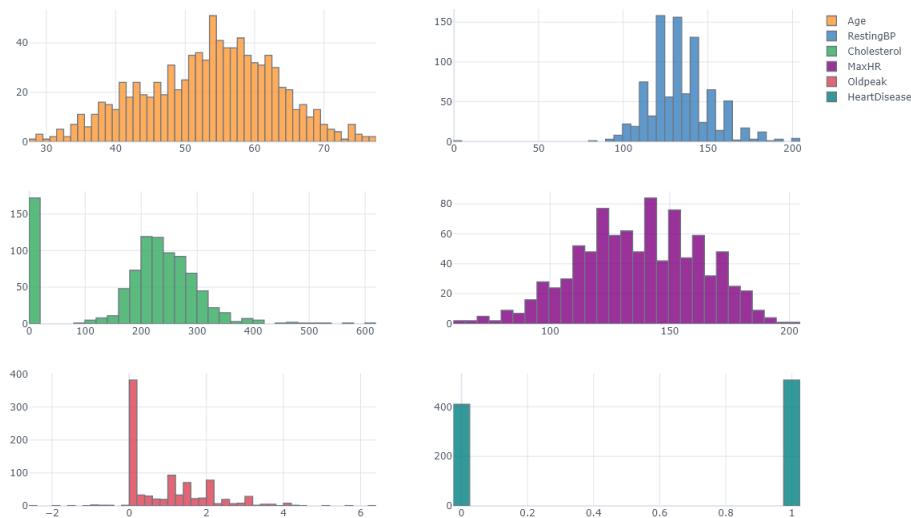




Таблица №12

ChestPainType			Healthy		Heart		Всего	
			N	%	N	%	N	%
ASY	Female	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	4	19.05	17	80.95	21	100.00
		<b>40 - &lt;65 years</b>	64	18.39	284	81.61	348	100.00
		<b>65 - &lt;75 years</b>	4	7.69	48	92.31	52	100.00
		<b>75+ years</b>	1	20.00	4	80.00	5	100.00
		<b>Всего</b>	73	17.14	353	82.86	426	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	3	50.00	3	50.00	6	100.00
		<b>40 - &lt;65 years</b>	26	44.07	33	55.93	59	100.00
		<b>65 - &lt;75 years</b>	2	40.00	3	60.00	5	100.00
		<b>75+ years</b>						
		<b>Всего</b>	31	44.29	39	55.71	70	100.00
	Total	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	7	25.93	20	74.07	27	100.00
		<b>40 - &lt;65 years</b>	90	22.11	317	77.89	407	100.00
		<b>65 - &lt;75 years</b>	6	10.53	51	89.47	57	100.00
		<b>75+ years</b>	1	20.00	4	80.00	5	100.00
		<b>Всего</b>	104	20.97	392	79.03	496	100.00
	Всего	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	14	25.93	40	74.07	54	100.00
		<b>40 - &lt;65 years</b>	180	22.11	634	77.89	814	100.00
		<b>65 - &lt;75 years</b>	12	10.53	102	89.47	114	100.00
		<b>75+ years</b>	2	20.00	8	80.00	10	100.00
		<b>Всего</b>	208	20.97	784	79.03	992	100.00



ChestPainType			Healthy		Heart		Всего	
			N	%	N	%	N	%
ATA	Female	<b>0 - 29 years</b>	4	100.00		0.00	4	100.00
		<b>30 - &lt;39 years</b>	15	88.24	2	11.76	17	100.00
		<b>40 - &lt;65 years</b>	73	82.02	16	17.98	89	100.00
		<b>65 - &lt;75 years</b>	1	33.33	2	66.67	3	100.00
		<b>75+ years</b>						
		<b>Всего</b>	93	82.30	20	17.70	113	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	6	100.00		0.00	6	100.00
		<b>40 - &lt;65 years</b>	48	92.31	4	7.69	52	100.00
		<b>65 - &lt;75 years</b>	2	100.00		0.00	2	100.00
		<b>75+ years</b>						
		<b>Всего</b>	56	93.33	4	6.67	60	100.00
	Total	<b>0 - 29 years</b>	4	100.00		0.00	4	100.00
		<b>30 - &lt;39 years</b>	21	91.30	2	8.70	23	100.00
		<b>40 - &lt;65 years</b>	121	85.82	20	14.18	141	100.00
		<b>65 - &lt;75 years</b>	3	60.00	2	40.00	5	100.00
		<b>75+ years</b>						
		<b>Всего</b>	149	86.13	24	13.87	173	100.00
	Всего	<b>0 - 29 years</b>	8	100.00		0.00	8	100.00
		<b>30 - &lt;39 years</b>	42	91.30	4	8.70	46	100.00
		<b>40 - &lt;65 years</b>	242	85.82	40	14.18	282	100.00
		<b>65 - &lt;75 years</b>	6	60.00	4	40.00	10	100.00
		<b>75+ years</b>						
		<b>Всего</b>	298	86.13	48	13.87	346	100.00



ChestPainType			Healthy		Heart		Всего	
			N	%	N	%	N	%
NAP	Female	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	14	93.33	1	6.67	15	100.00
		<b>40 - &lt;65 years</b>	63	53.39	55	46.61	118	100.00
		<b>65 - &lt;75 years</b>	7	43.75	9	56.25	16	100.00
		<b>75+ years</b>		0.00	1	100.00	1	100.00
		<b>Всего</b>	84	56.00	66	44.00	150	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	5	100.00		0.00	5	100.00
		<b>40 - &lt;65 years</b>	33	86.84	5	13.16	38	100.00
		<b>65 - &lt;75 years</b>	8	88.89	1	11.11	9	100.00
		<b>75+ years</b>	1	100.00		0.00	1	100.00
		<b>Всего</b>	47	88.68	6	11.32	53	100.00
	Total	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	19	95.00	1	5.00	20	100.00
		<b>40 - &lt;65 years</b>	96	61.54	60	38.46	156	100.00
		<b>65 - &lt;75 years</b>	15	60.00	10	40.00	25	100.00
		<b>75+ years</b>	1	50.00	1	50.00	2	100.00
		<b>Всего</b>	131	64.53	72	35.47	203	100.00
	Всего	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	38	95.00	2	5.00	40	100.00
		<b>40 - &lt;65 years</b>	192	61.54	120	38.46	312	100.00
		<b>65 - &lt;75 years</b>	30	60.00	20	40.00	50	100.00
		<b>75+ years</b>	2	50.00	2	50.00	4	100.00
		<b>Всего</b>	262	64.53	144	35.47	406	100.00



ChestPainType			Healthy		Heart		Всего	
			N	%	N	%	N	%
TA	Female	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	1	25.00	3	75.00	4	100.00
		<b>40 - &lt;65 years</b>	13	52.00	12	48.00	25	100.00
		<b>65 - &lt;75 years</b>	3	42.86	4	57.14	7	100.00
		<b>75+ years</b>						
		<b>Всего</b>	17	47.22	19	52.78	36	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	2	100.00		0.00	2	100.00
		<b>40 - &lt;65 years</b>	5	83.33	1	16.67	6	100.00
		<b>65 - &lt;75 years</b>	2	100.00		0.00	2	100.00
		<b>75+ years</b>						
		<b>Всего</b>	9	90.00	1	10.00	10	100.00
	Total	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	3	50.00	3	50.00	6	100.00
		<b>40 - &lt;65 years</b>	18	58.06	13	41.94	31	100.00
		<b>65 - &lt;75 years</b>	5	55.56	4	44.44	9	100.00
		<b>75+ years</b>						
		<b>Всего</b>	26	56.52	20	43.48	46	100.00
	Всего	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	6	50.00	6	50.00	12	100.00
		<b>40 - &lt;65 years</b>	36	58.06	26	41.94	62	100.00
		<b>65 - &lt;75 years</b>	10	55.56	8	44.44	18	100.00
		<b>75+ years</b>						
		<b>Всего</b>	52	56.52	40	43.48	92	100.00



ChestPainType			Healthy		Heart		Всего	
			N	%	N	%	N	%
Всего	Female	0 - 29 years	4	100.00		0.00	4	100.00
		30 - <39 years	34	59.65	23	40.35	57	100.00
		40 - <65 years	213	36.72	367	63.28	580	100.00
		65 - <75 years	15	19.23	63	80.77	78	100.00
		75+ years	1	16.67	5	83.33	6	100.00
		Всего	267	36.83	458	63.17	725	100.00
		0 - 29 years						
	Male	30 - <39 years	16	84.21	3	15.79	19	100.00
		40 - <65 years	112	72.26	43	27.74	155	100.00
		65 - <75 years	14	77.78	4	22.22	18	100.00
		75+ years	1	100.00		0.00	1	100.00
		Всего	143	74.09	50	25.91	193	100.00
		0 - 29 years						
		Total	0 - 29 years	4	100.00		0.00	4
	Всего	30 - <39 years	50	65.79	26	34.21	76	100.00
		40 - <65 years	325	44.22	410	55.78	735	100.00
		65 - <75 years	29	30.21	67	69.79	96	100.00
		75+ years	2	28.57	5	71.43	7	100.00
		Всего	410	44.66	508	55.34	918	100.00
		0 - 29 years						
		0 - 29 years	8	100.00		0.00	8	100.00
		30 - <39 years	100	65.79	52	34.21	152	100.00
		40 - <65 years	650	44.22	820	55.78	1470	100.00
		65 - <75 years	58	30.21	134	69.79	192	100.00
		75+ years	4	28.57	10	71.43	14	100.00
		Всего	820	44.66	1016	55.34	1836	100.00

© Dr. Alexander Wagner. Все права охраняются законом



## График №11. Распределение переменной Cholesterol по классу (Age, Sex, FastingBS) в виде 4 субграфиков на одной панели

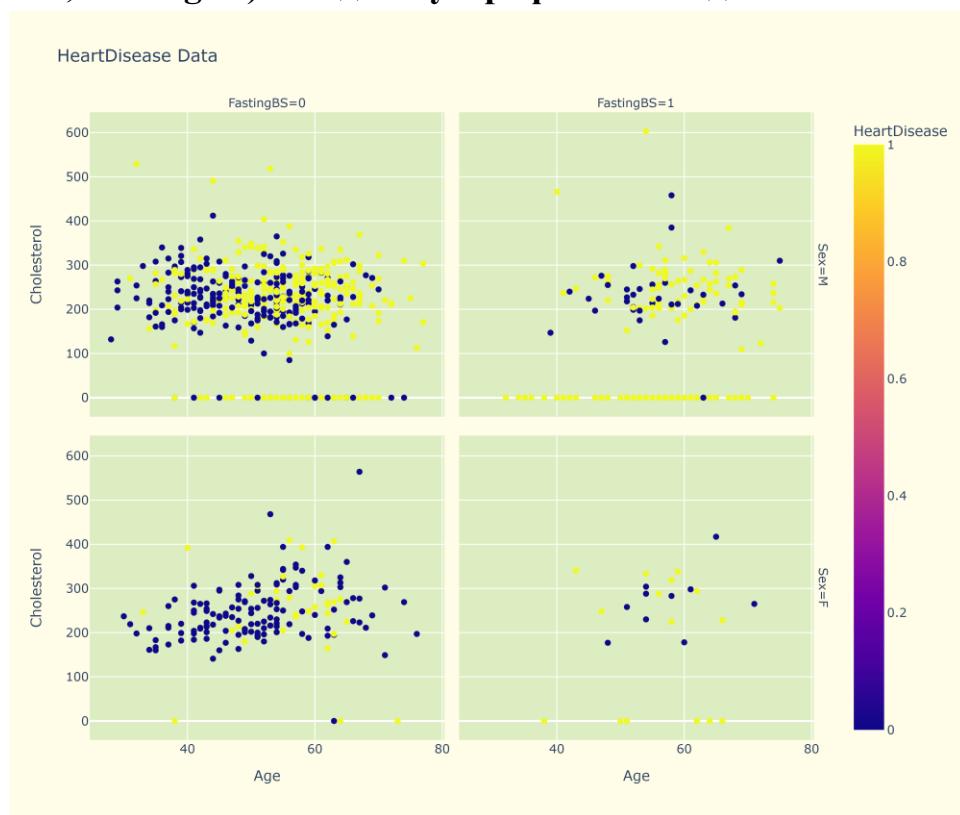




Таблица №13

ExerciseAngina			Healthy		Heart		Всего	
			N	%	N	%	N	%
N	Female	<b>0 - 29 years</b>	4	100.00		0.00	4	100.00
		<b>30 - &lt;39 years</b>	33	75.00	11	25.00	44	100.00
		<b>40 - &lt;65 years</b>	178	57.23	133	42.77	311	100.00
		<b>65 - &lt;75 years</b>	13	36.11	23	63.89	36	100.00
		<b>75+ years</b>		0.00	2	100.00	2	100.00
		<b>Всего</b>	228	57.43	169	42.57	397	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	16	88.89	2	11.11	18	100.00
		<b>40 - &lt;65 years</b>	97	84.35	18	15.65	115	100.00
		<b>65 - &lt;75 years</b>	13	81.25	3	18.75	16	100.00
		<b>75+ years</b>	1	100.00		0.00	1	100.00
		<b>Всего</b>	127	84.67	23	15.33	150	100.00
	Total	<b>0 - 29 years</b>	4	100.00		0.00	4	100.00
		<b>30 - &lt;39 years</b>	49	79.03	13	20.97	62	100.00
		<b>40 - &lt;65 years</b>	275	64.55	151	35.45	426	100.00
		<b>65 - &lt;75 years</b>	26	50.00	26	50.00	52	100.00
		<b>75+ years</b>	1	33.33	2	66.67	3	100.00
		<b>Всего</b>	355	64.90	192	35.10	547	100.00
	Всего	<b>0 - 29 years</b>	8	100.00		0.00	8	100.00
		<b>30 - &lt;39 years</b>	98	79.03	26	20.97	124	100.00
		<b>40 - &lt;65 years</b>	550	64.55	302	35.45	852	100.00
		<b>65 - &lt;75 years</b>	52	50.00	52	50.00	104	100.00
		<b>75+ years</b>	2	33.33	4	66.67	6	100.00
		<b>Всего</b>	710	64.90	384	35.10	1094	100.00



ExerciseAngina			Healthy		Heart		Всего	
			N	%	N	%	N	%
Y	Female	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	1	7.69	12	92.31	13	100.00
		<b>40 - &lt;65 years</b>	35	13.01	234	86.99	269	100.00
		<b>65 - &lt;75 years</b>	2	4.76	40	95.24	42	100.00
		<b>75+ years</b>	1	25.00	3	75.00	4	100.00
		<b>Всего</b>	39	11.89	289	88.11	328	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>		0.00	1	100.00	1	100.00
		<b>40 - &lt;65 years</b>	15	37.50	25	62.50	40	100.00
		<b>65 - &lt;75 years</b>	1	50.00	1	50.00	2	100.00
		<b>75+ years</b>						
		<b>Всего</b>	16	37.21	27	62.79	43	100.00
	Total	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	1	7.14	13	92.86	14	100.00
		<b>40 - &lt;65 years</b>	50	16.18	259	83.82	309	100.00
		<b>65 - &lt;75 years</b>	3	6.82	41	93.18	44	100.00
		<b>75+ years</b>	1	25.00	3	75.00	4	100.00
		<b>Всего</b>	55	14.82	316	85.18	371	100.00
	Всего	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	2	7.14	26	92.86	28	100.00
		<b>40 - &lt;65 years</b>	100	16.18	518	83.82	618	100.00
		<b>65 - &lt;75 years</b>	6	6.82	82	93.18	88	100.00
		<b>75+ years</b>	2	25.00	6	75.00	8	100.00
		<b>Всего</b>	110	14.82	632	85.18	742	100.00



ExerciseAngina			Healthy		Heart		Всего	
			N	%	N	%	N	%
Всего	Female	0 - 29 years	4	100.00		0.00	4	100.00
		30 - <39 years	34	59.65	23	40.35	57	100.00
		40 - <65 years	213	36.72	367	63.28	580	100.00
		65 - <75 years	15	19.23	63	80.77	78	100.00
		75+ years	1	16.67	5	83.33	6	100.00
	Male	Всего	267	36.83	458	63.17	725	100.00
		0 - 29 years						
		30 - <39 years	16	84.21	3	15.79	19	100.00
		40 - <65 years	112	72.26	43	27.74	155	100.00
		65 - <75 years	14	77.78	4	22.22	18	100.00
	Total	75+ years	1	100.00		0.00	1	100.00
		Всего	143	74.09	50	25.91	193	100.00
		0 - 29 years	4	100.00		0.00	4	100.00
		30 - <39 years	50	65.79	26	34.21	76	100.00
		40 - <65 years	325	44.22	410	55.78	735	100.00
	Всего	65 - <75 years	29	30.21	67	69.79	96	100.00
		75+ years	2	28.57	5	71.43	7	100.00
		Всего	410	44.66	508	55.34	918	100.00
		0 - 29 years	8	100.00		0.00	8	100.00
		30 - <39 years	100	65.79	52	34.21	152	100.00
	Всего	40 - <65 years	650	44.22	820	55.78	1470	100.00
		65 - <75 years	58	30.21	134	69.79	192	100.00
		75+ years	4	28.57	10	71.43	14	100.00
		Всего	820	44.66	1016	55.34	1836	100.00

© Dr. Alexander Wagner. Все права охраняются законом



График №12. Распределение по возрасту (Age) для переменных: 'Sex','ChestPainType','FastingBS','RestingECG','ExerciseAngina','ST\_Slope','HeartDisease' в форме Виалин-графиков

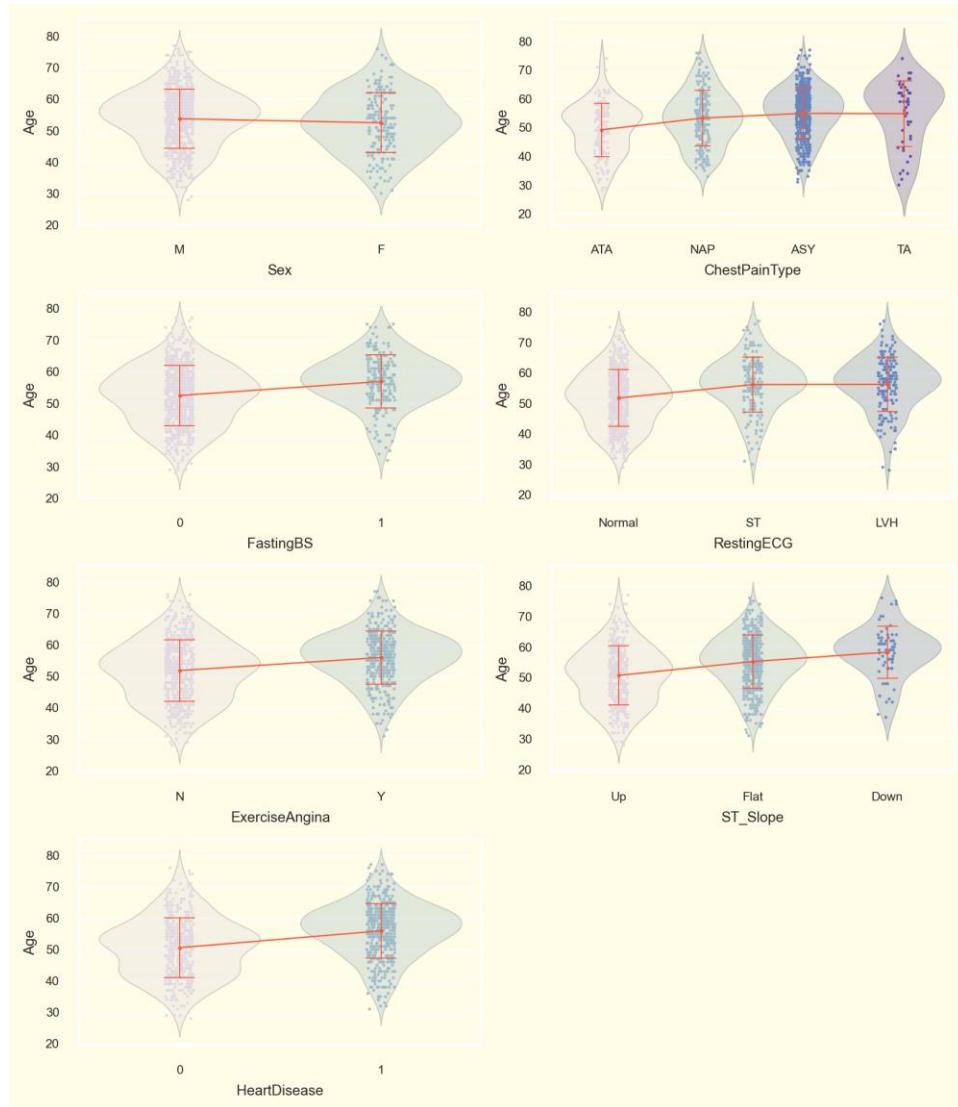




Таблица №14

RestingECG			Healthy		Heart		Всего	
			N	%	N	%	N	%
LVH	Female	<b>0 - 29 years</b>	2	100.00		0.00	2	100.00
		<b>30 - &lt;39 years</b>	4	80.00	1	20.00	5	100.00
		<b>40 - &lt;65 years</b>	36	32.73	74	67.27	110	100.00
		<b>65 - &lt;75 years</b>	7	31.82	15	68.18	22	100.00
		<b>75+ years</b>		0.00	2	100.00	2	100.00
		<b>Всего</b>	49	34.75	92	65.25	141	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>						
		<b>40 - &lt;65 years</b>	26	66.67	13	33.33	39	100.00
		<b>65 - &lt;75 years</b>	7	87.50	1	12.50	8	100.00
		<b>75+ years</b>						
		<b>Всего</b>	33	70.21	14	29.79	47	100.00
	Total	<b>0 - 29 years</b>	2	100.00		0.00	2	100.00
		<b>30 - &lt;39 years</b>	4	80.00	1	20.00	5	100.00
		<b>40 - &lt;65 years</b>	62	41.61	87	58.39	149	100.00
		<b>65 - &lt;75 years</b>	14	46.67	16	53.33	30	100.00
		<b>75+ years</b>		0.00	2	100.00	2	100.00
		<b>Всего</b>	82	43.62	106	56.38	188	100.00
	Всего	<b>0 - 29 years</b>	4	100.00		0.00	4	100.00
		<b>30 - &lt;39 years</b>	8	80.00	2	20.00	10	100.00
		<b>40 - &lt;65 years</b>	124	41.61	174	58.39	298	100.00
		<b>65 - &lt;75 years</b>	28	46.67	32	53.33	60	100.00
		<b>75+ years</b>		0.00	4	100.00	4	100.00
		<b>Всего</b>	164	43.62	212	56.38	376	100.00



RestingECG			Healthy		Heart		Всего	
			N	%	N	%	N	%
Normal	Female	<b>0 - 29 years</b>	2	100.00		0.00	2	100.00
		<b>30 - &lt;39 years</b>	26	54.17	22	45.83	48	100.00
		<b>40 - &lt;65 years</b>	143	41.33	203	58.67	346	100.00
		<b>65 - &lt;75 years</b>	6	16.67	30	83.33	36	100.00
		<b>75+ years</b>	1	50.00	1	50.00	2	100.00
		<b>Всего</b>	178	41.01	256	58.99	434	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	11	78.57	3	21.43	14	100.00
		<b>40 - &lt;65 years</b>	71	74.74	24	25.26	95	100.00
		<b>65 - &lt;75 years</b>	7	77.78	2	22.22	9	100.00
		<b>75+ years</b>						
		<b>Всего</b>	89	75.42	29	24.58	118	100.00
	Total	<b>0 - 29 years</b>	2	100.00		0.00	2	100.00
		<b>30 - &lt;39 years</b>	37	59.68	25	40.32	62	100.00
		<b>40 - &lt;65 years</b>	214	48.53	227	51.47	441	100.00
		<b>65 - &lt;75 years</b>	13	28.89	32	71.11	45	100.00
		<b>75+ years</b>	1	50.00	1	50.00	2	100.00
		<b>Всего</b>	267	48.37	285	51.63	552	100.00
	Всего	<b>0 - 29 years</b>	4	100.00		0.00	4	100.00
		<b>30 - &lt;39 years</b>	74	59.68	50	40.32	124	100.00
		<b>40 - &lt;65 years</b>	428	48.53	454	51.47	882	100.00
		<b>65 - &lt;75 years</b>	26	28.89	64	71.11	90	100.00
		<b>75+ years</b>	2	50.00	2	50.00	4	100.00
		<b>Всего</b>	534	48.37	570	51.63	1104	100.00



RestingECG			Healthy		Heart		Всего	
			N	%	N	%	N	%
ST	Female	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	4	100.00		0.00	4	100.00
		<b>40 - &lt;65 years</b>	34	27.42	90	72.58	124	100.00
		<b>65 - &lt;75 years</b>	2	10.00	18	90.00	20	100.00
		<b>75+ years</b>		0.00	2	100.00	2	100.00
		<b>Всего</b>	40	26.67	110	73.33	150	100.00
	Male	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	5	100.00		0.00	5	100.00
		<b>40 - &lt;65 years</b>	15	71.43	6	28.57	21	100.00
		<b>65 - &lt;75 years</b>		0.00	1	100.00	1	100.00
		<b>75+ years</b>	1	100.00		0.00	1	100.00
		<b>Всего</b>	21	75.00	7	25.00	28	100.00
	Total	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	9	100.00		0.00	9	100.00
		<b>40 - &lt;65 years</b>	49	33.79	96	66.21	145	100.00
		<b>65 - &lt;75 years</b>	2	9.52	19	90.48	21	100.00
		<b>75+ years</b>	1	33.33	2	66.67	3	100.00
		<b>Всего</b>	61	34.27	117	65.73	178	100.00
	Всего	<b>0 - 29 years</b>						
		<b>30 - &lt;39 years</b>	18	100.00		0.00	18	100.00
		<b>40 - &lt;65 years</b>	98	33.79	192	66.21	290	100.00
		<b>65 - &lt;75 years</b>	4	9.52	38	90.48	42	100.00
		<b>75+ years</b>	2	33.33	4	66.67	6	100.00
		<b>Всего</b>	122	34.27	234	65.73	356	100.00

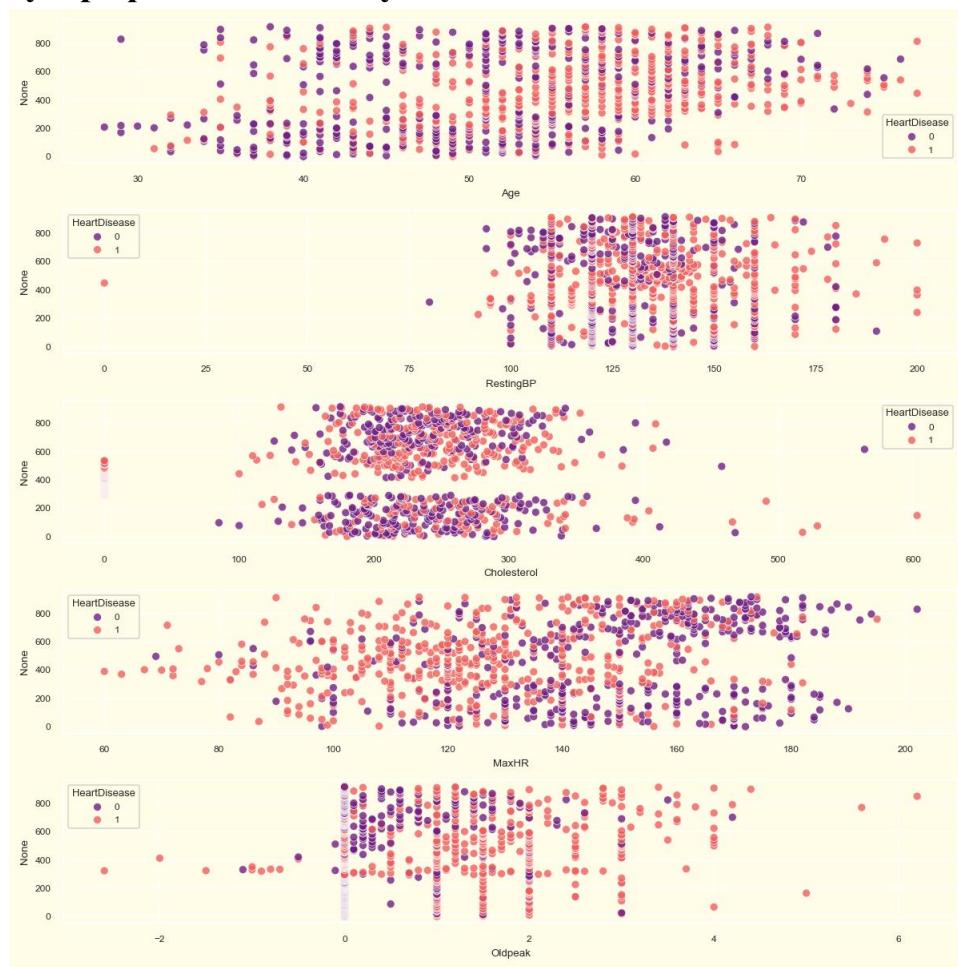


RestingECG			Healthy		Heart		Всего	
			N	%	N	%	N	%
Всего	Female	0 - 29 years	4	100.00		0.00	4	100.00
		30 - <39 years	34	59.65	23	40.35	57	100.00
		40 - <65 years	213	36.72	367	63.28	580	100.00
		65 - <75 years	15	19.23	63	80.77	78	100.00
		75+ years	1	16.67	5	83.33	6	100.00
		Всего	267	36.83	458	63.17	725	100.00
		0 - 29 years						
	Male	30 - <39 years	16	84.21	3	15.79	19	100.00
		40 - <65 years	112	72.26	43	27.74	155	100.00
		65 - <75 years	14	77.78	4	22.22	18	100.00
		75+ years	1	100.00		0.00	1	100.00
		Всего	143	74.09	50	25.91	193	100.00
		0 - 29 years						
		Total	0 - 29 years	4	100.00		0.00	4
	Всего	30 - <39 years	50	65.79	26	34.21	76	100.00
		40 - <65 years	325	44.22	410	55.78	735	100.00
		65 - <75 years	29	30.21	67	69.79	96	100.00
		75+ years	2	28.57	5	71.43	7	100.00
		Всего	410	44.66	508	55.34	918	100.00
		0 - 29 years						
		0 - 29 years	8	100.00		0.00	8	100.00
		30 - <39 years	100	65.79	52	34.21	152	100.00
		40 - <65 years	650	44.22	820	55.78	1470	100.00
		65 - <75 years	58	30.21	134	69.79	192	100.00
		75+ years	4	28.57	10	71.43	14	100.00
		Всего	820	44.66	1016	55.34	1836	100.00

© Dr. Alexander Wagner. Все права охраняются законом

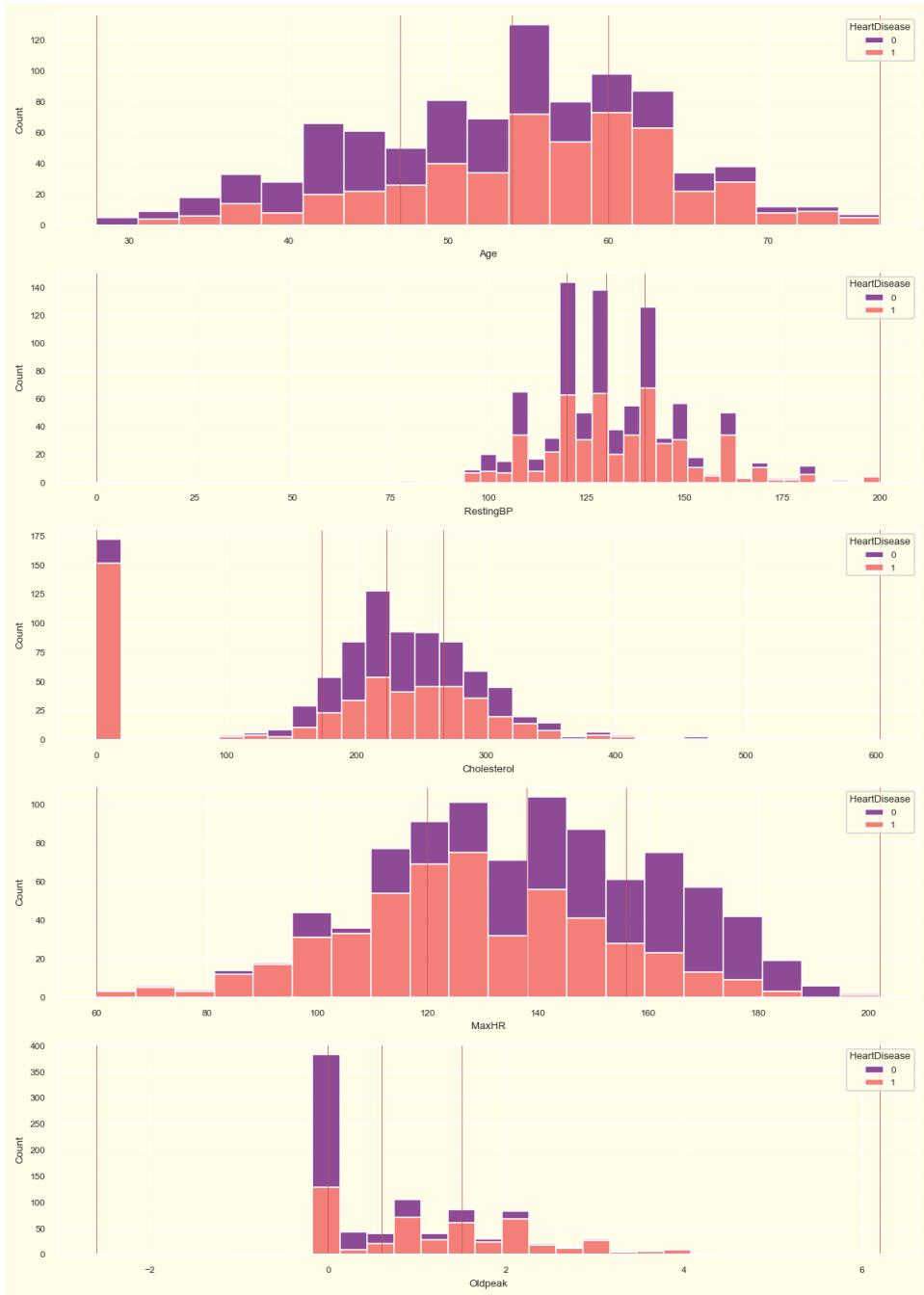


**График №13. Распределение всех числовых переменных в виде субграфиков по классу 'HeartDisease' на одной панели**



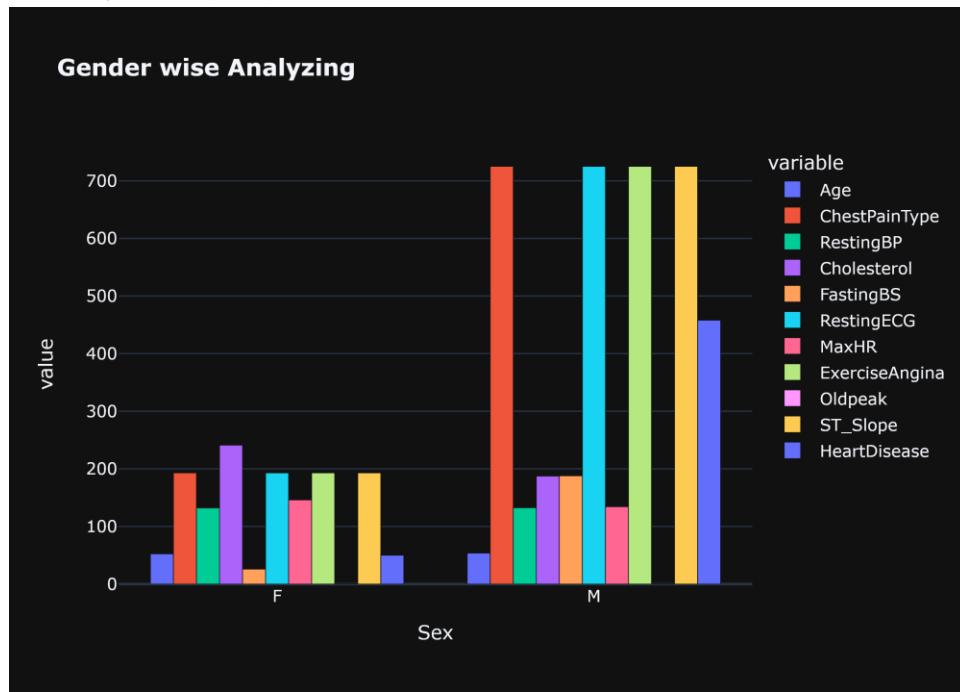


**График №14. Распределение всех числовых переменных в виде столбиковых диаграмм как субграфиков по классу 'HeartDisease' на одной панели**



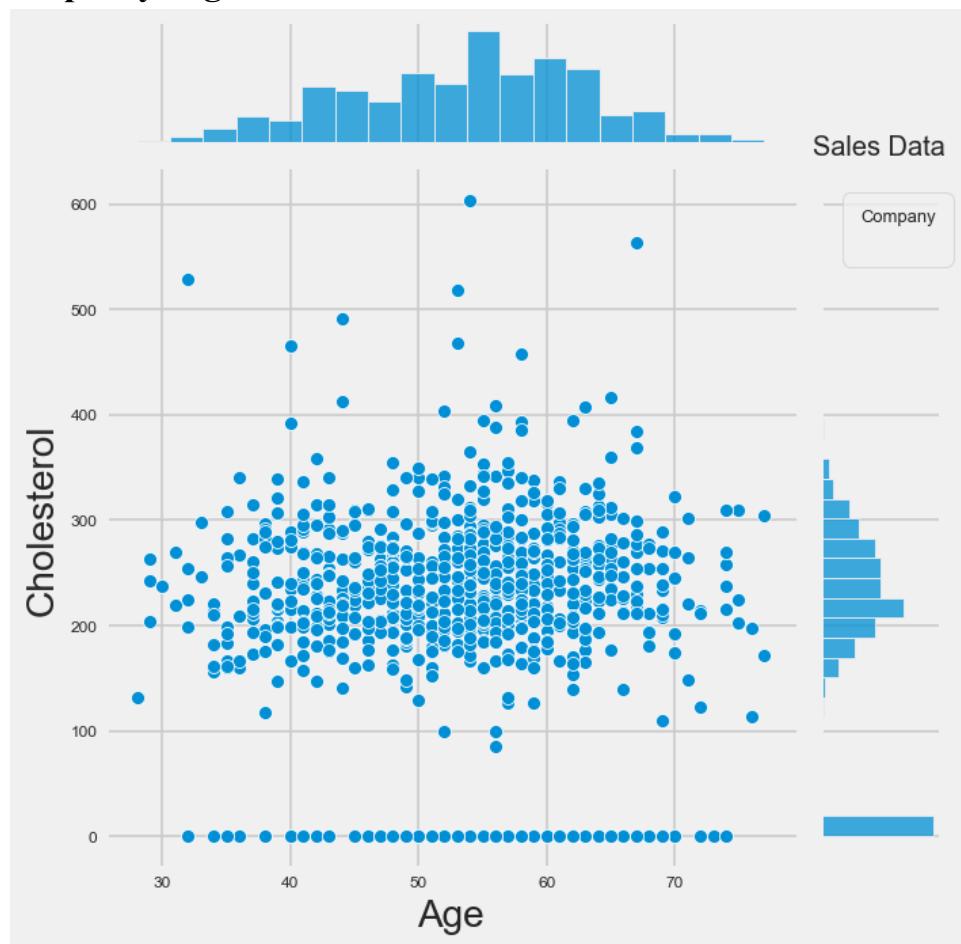


**График №15. Распределение численности пациентов по всем переменным в виде столбиковых диаграмм как субграфиков по классу пол (Sex) на одной панели**





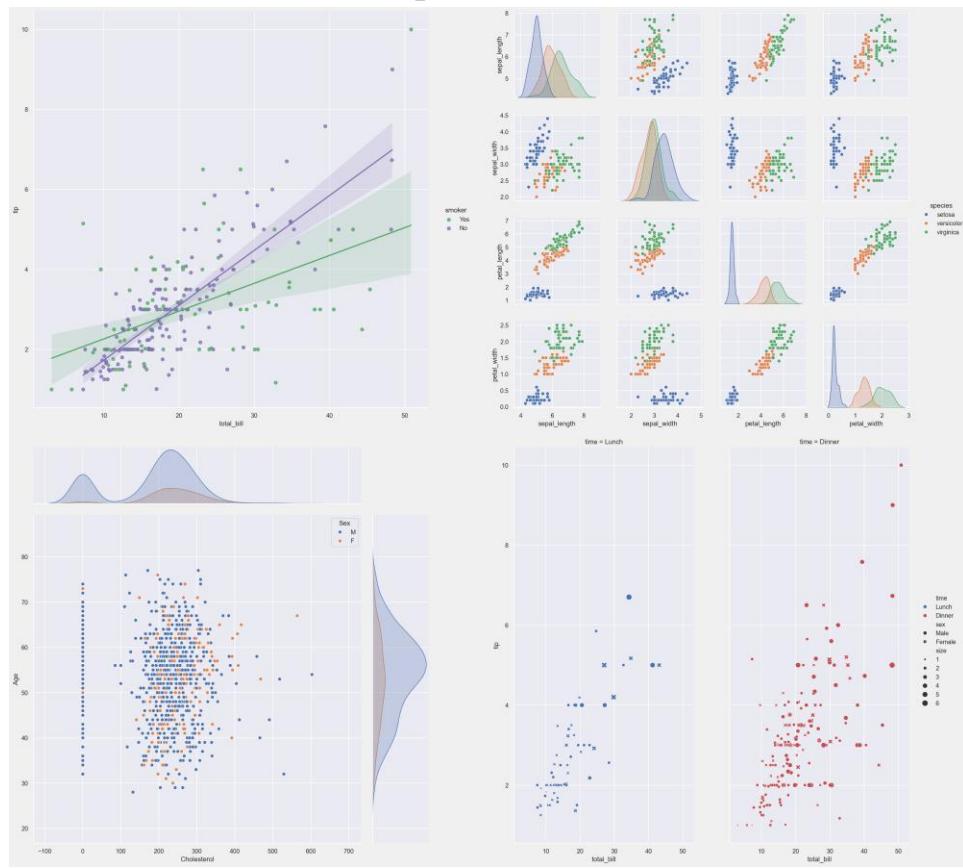
**График №16. Биполярное распределение переменной Cholesterol по возрасту 'Age'**







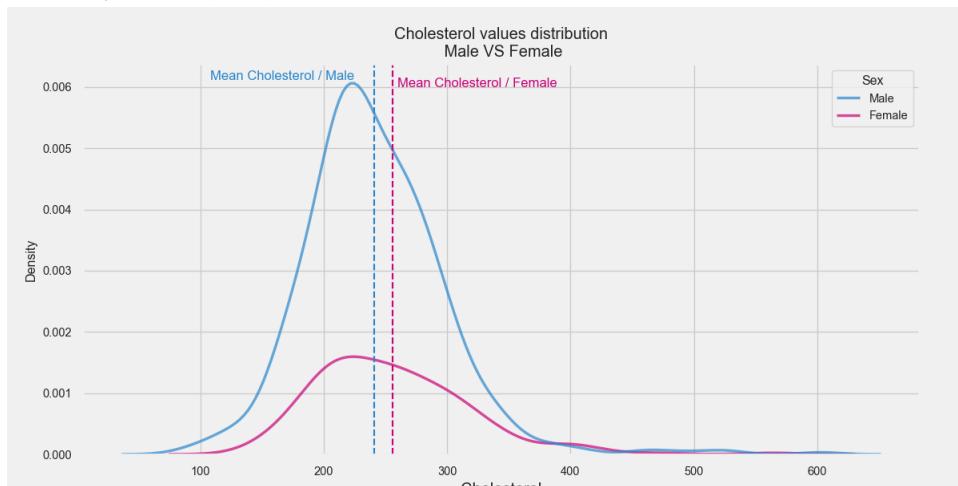
**График №17. Биполярное распределение разного графического типа всех численных переменных на одной панели**







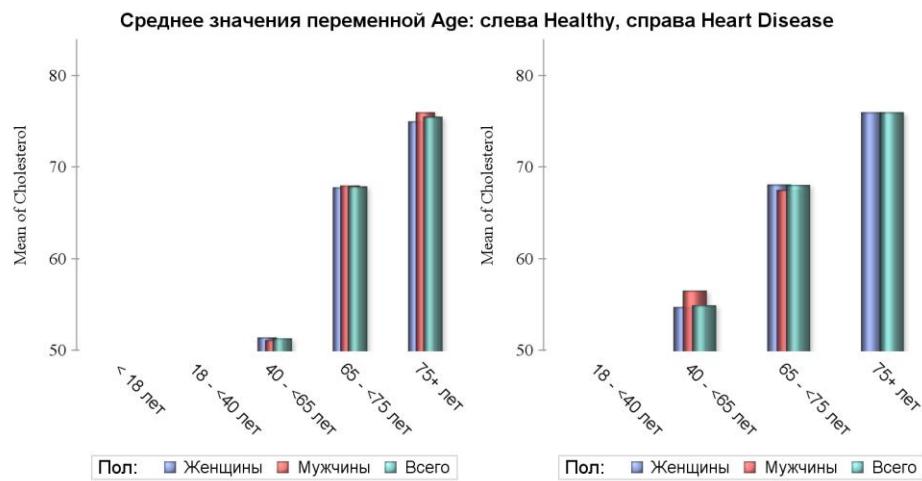
## График №18. Плотность распределения переменной Cholesterol по классу пол (Sex)







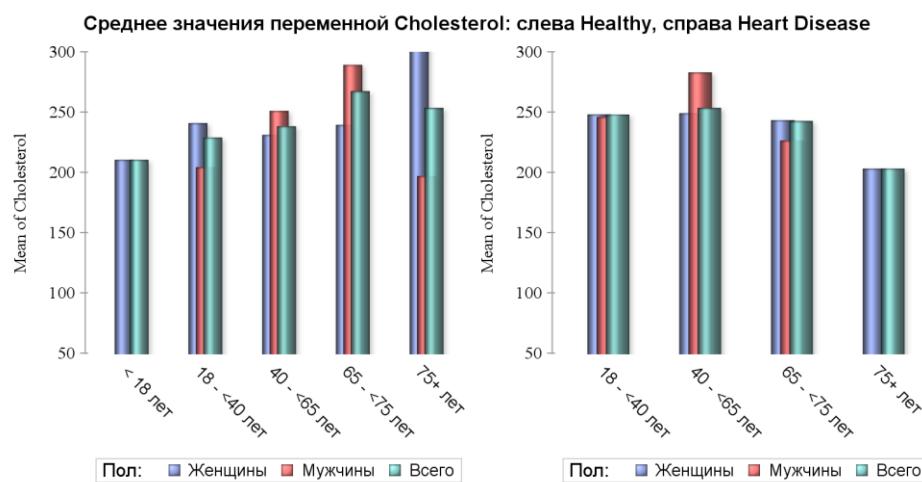
### График №19. Распределение переменной Age возрастным группам в виде столбиковых диаграмм по классу 'HeartDisease'







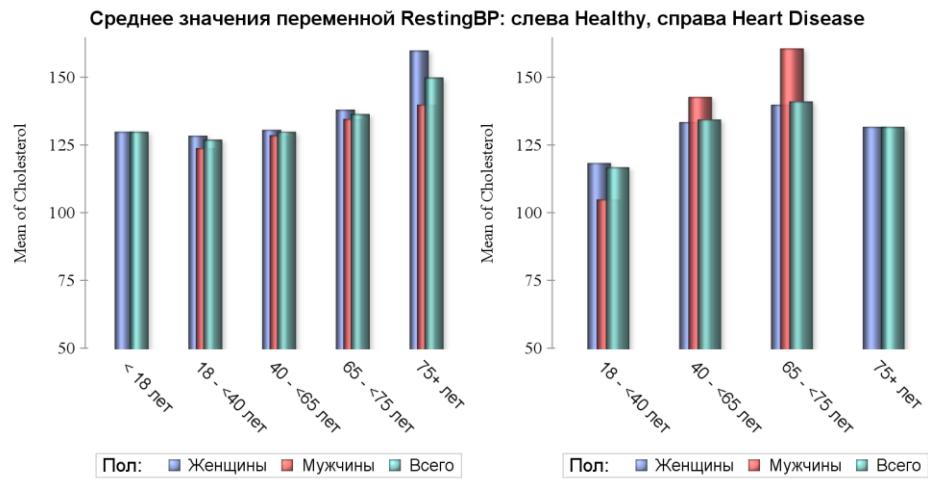
## График №20. Распределение переменной Cholesterol по возрастным группам в виде столбиковых диаграмм по классу 'HeartDisease'







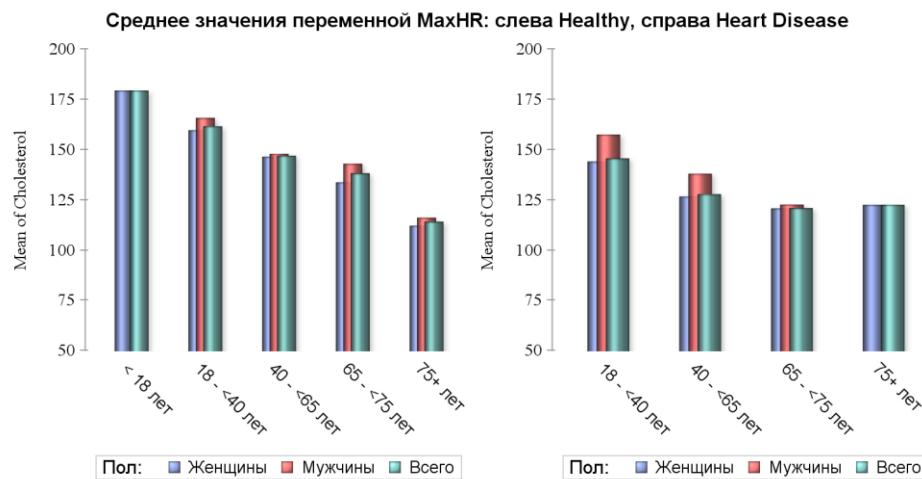
## График №21. Распределение переменной RestingBP по возрастным группам в виде столбиковых диаграмм по классу 'HeartDisease'







## График №22. Распределение переменной MaxHR по возрастным группам в виде столбиковых диаграмм по классу 'HeartDisease'







### График №23. Распределение переменной Oldpeak по возрастным группам в виде столбиковых диаграмм по классу 'HeartDisease'







**График № 24. Торт-диаграмма распределения пациентов по классу переменной HeartDisease**

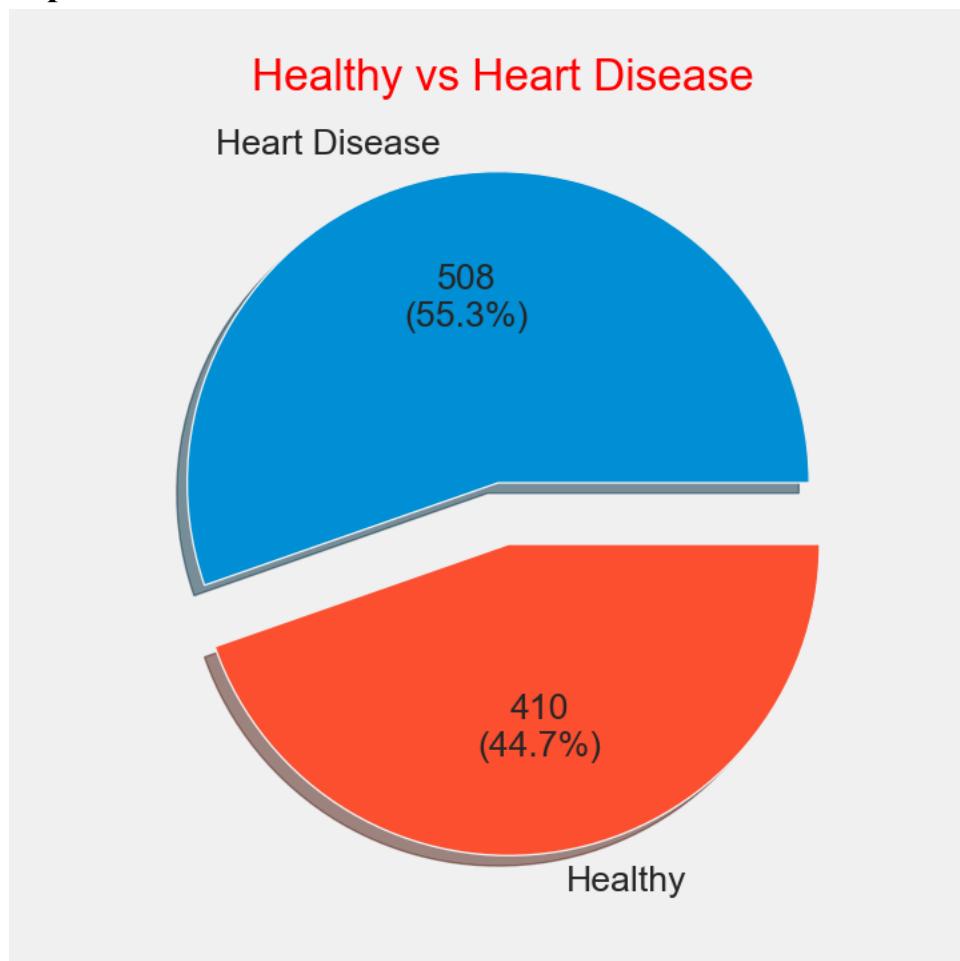
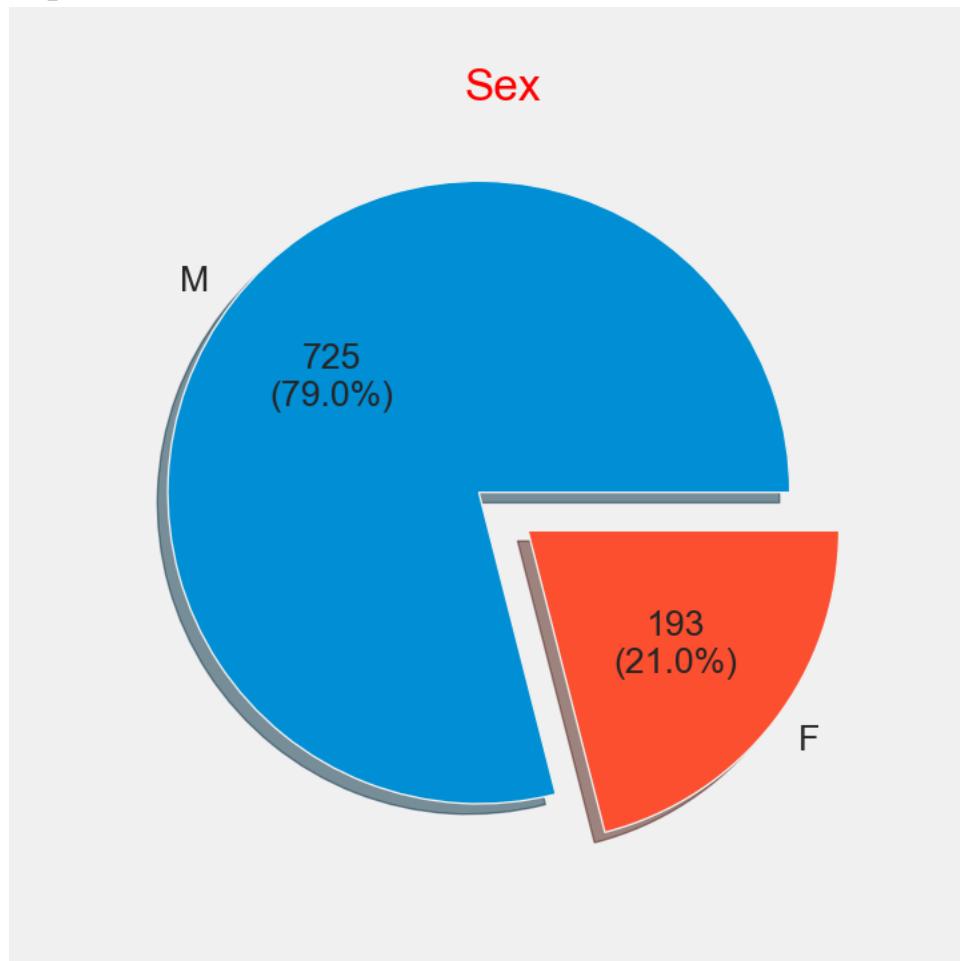






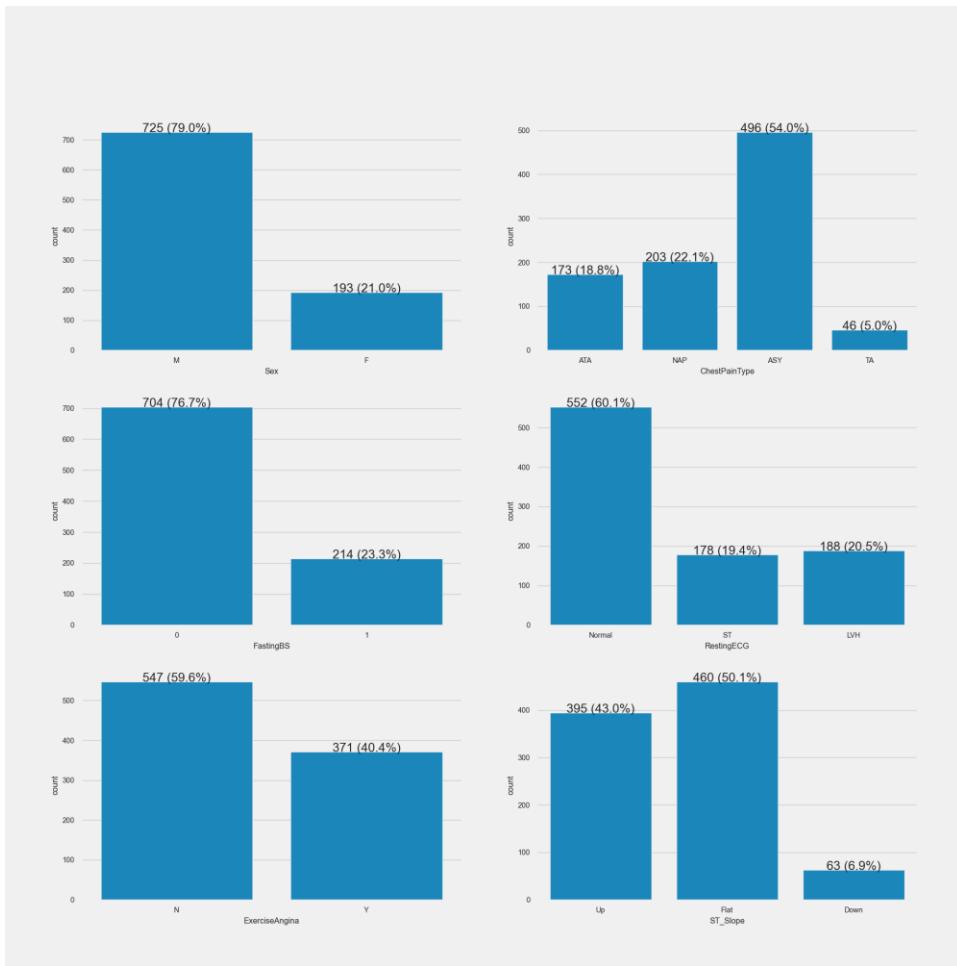
График №25. Торт-диаграмма распределения пациентов по классу-переменной Sex







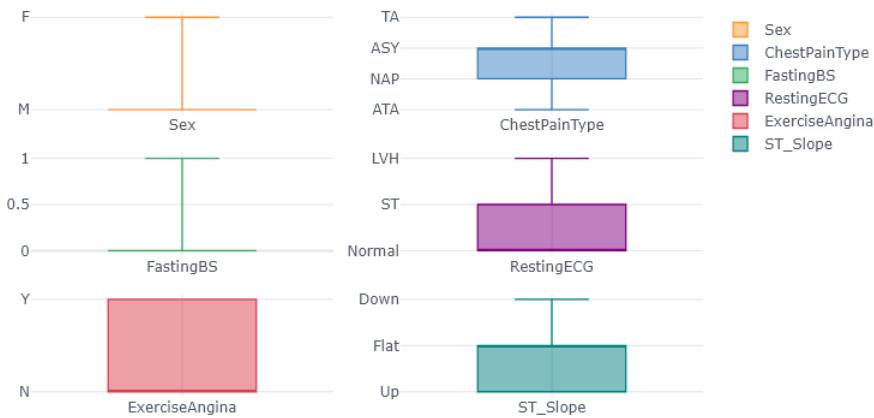
**График №26. Столбиковая диаграмма распределения переменных: 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST\_Slope', 'HeartDisease' по категориям**







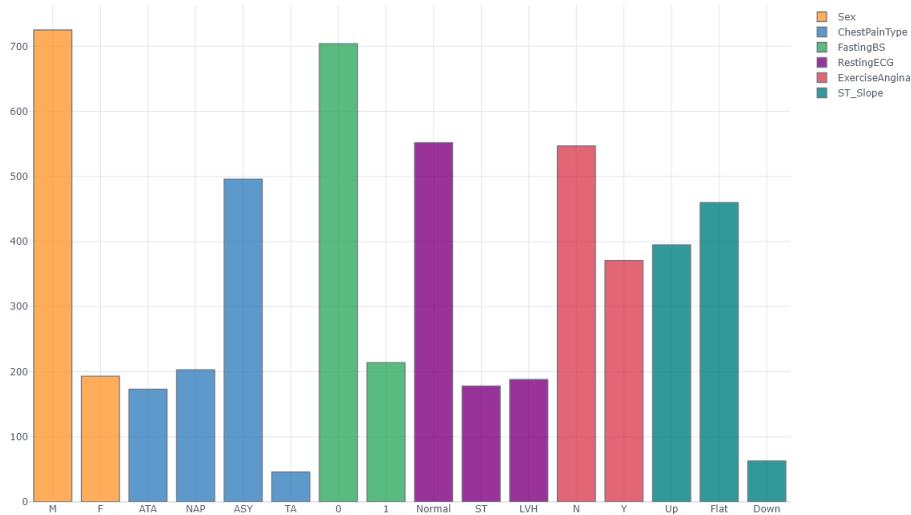
**График №27. Блок-бокс диаграмма распределения переменных: 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST\_Slope', 'HeartDisease' в виде субграфиков на одной панели по категориям**







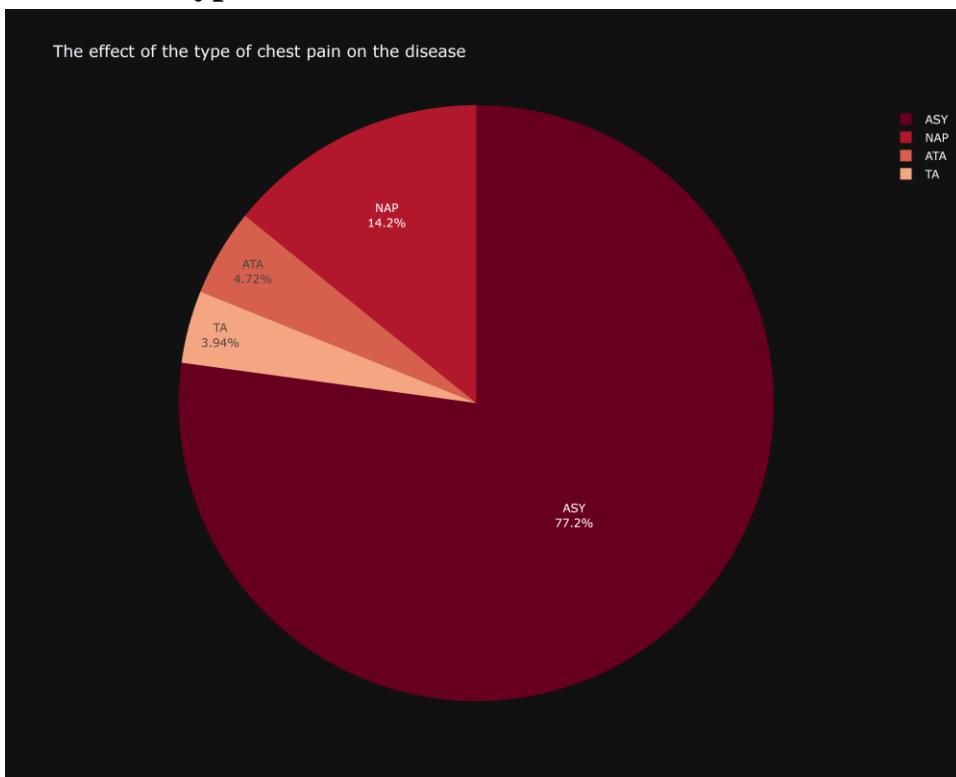
**График №28. Столбиковая диаграмма распределения переменных: 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST\_Slope', 'HeartDisease' по категориям**







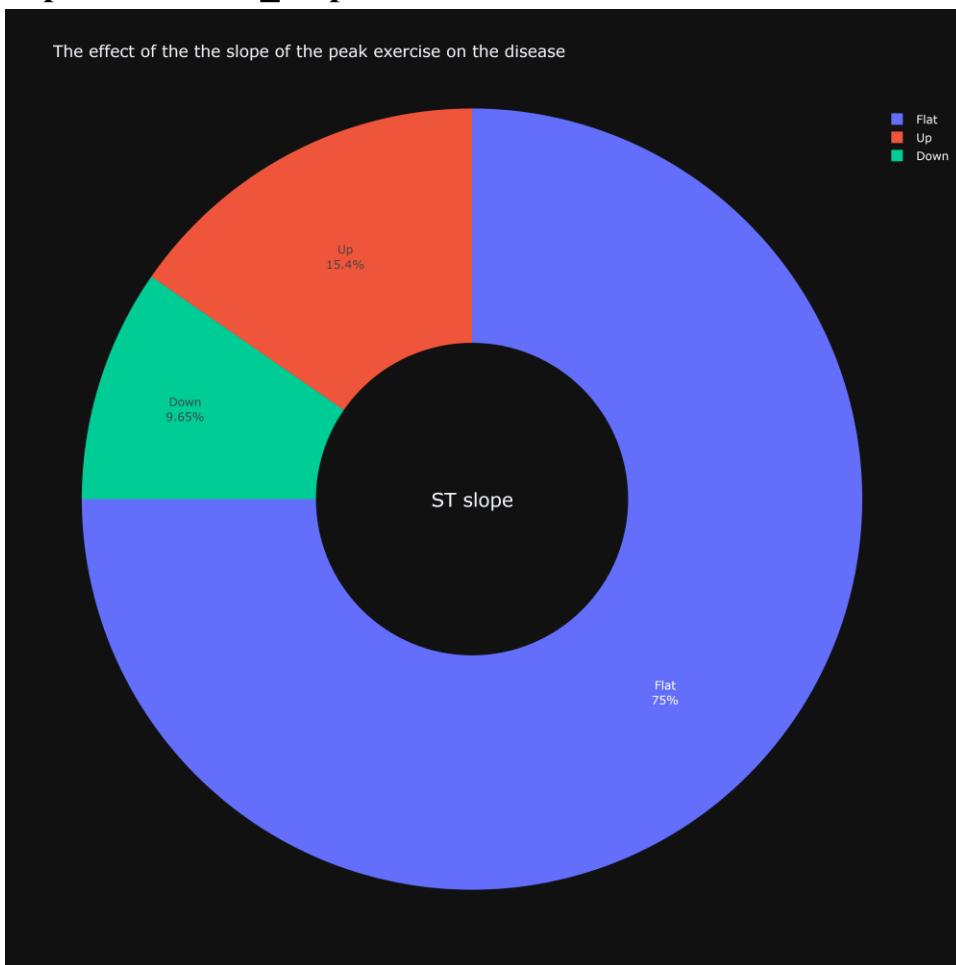
**График №29. Секторная диаграмма распределения переменной ChestPainType**







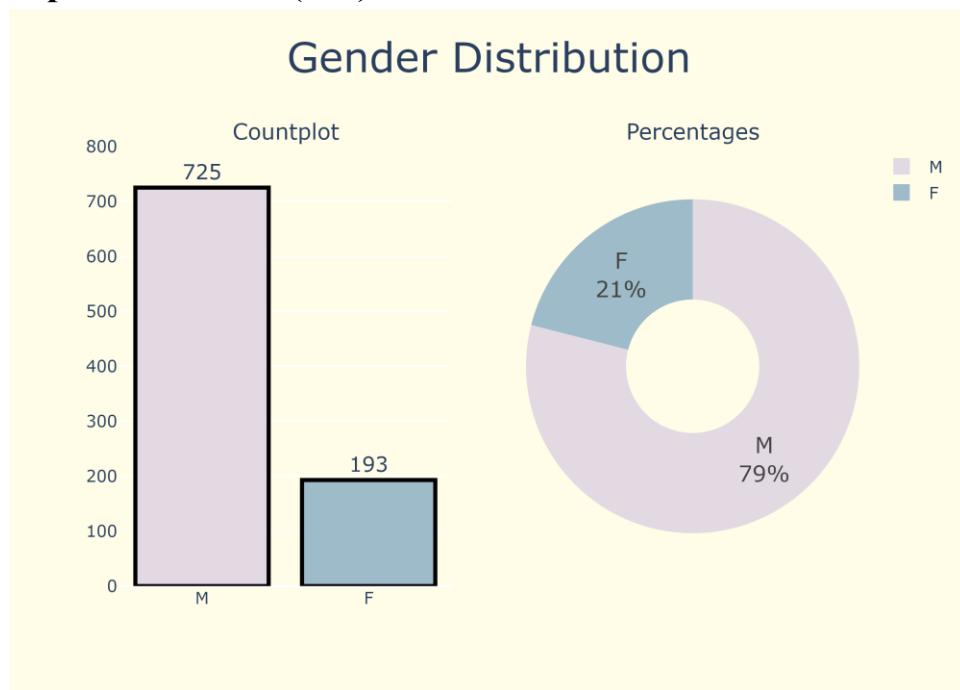
**График №30. Секторная диаграмма (2-го типа) распределения переменной ST\_Slope**







**График №31. Комбинированная диаграмма распределения переменной пол (Sex)**







**График №32. Комбинированная диаграмма распределения переменной HeartDisease**

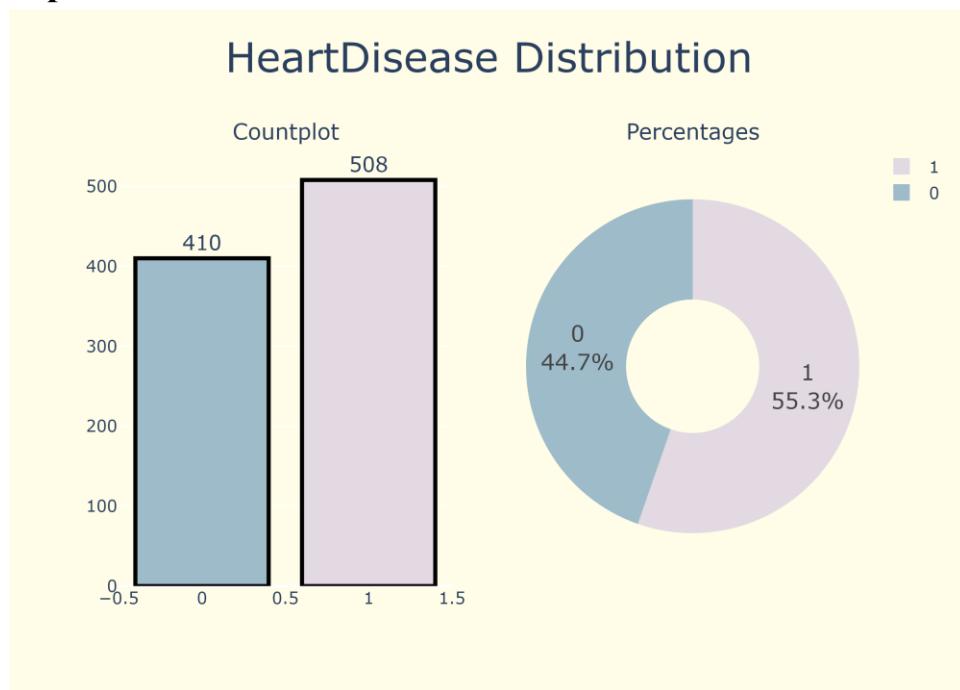
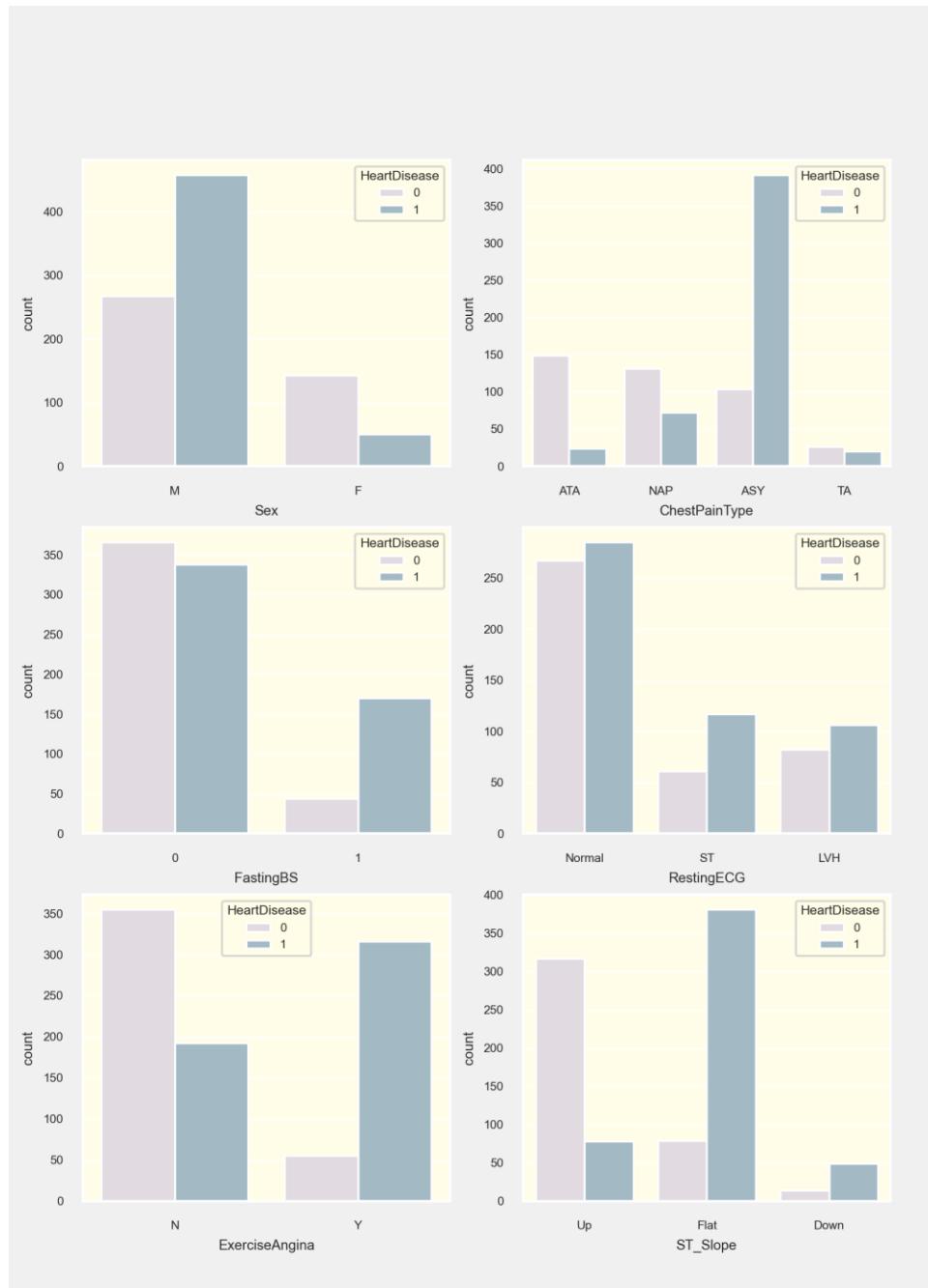






График №33. Столбиковая диаграмма распределения переменных: 'Sex', 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST\_Slope' по категориям и классу 'HeartDisease'







**График №34. Столбиковая диаграмма распределения переменных: 'ChestPainType', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST\_Slope' по категориям и классу 'Sex'**

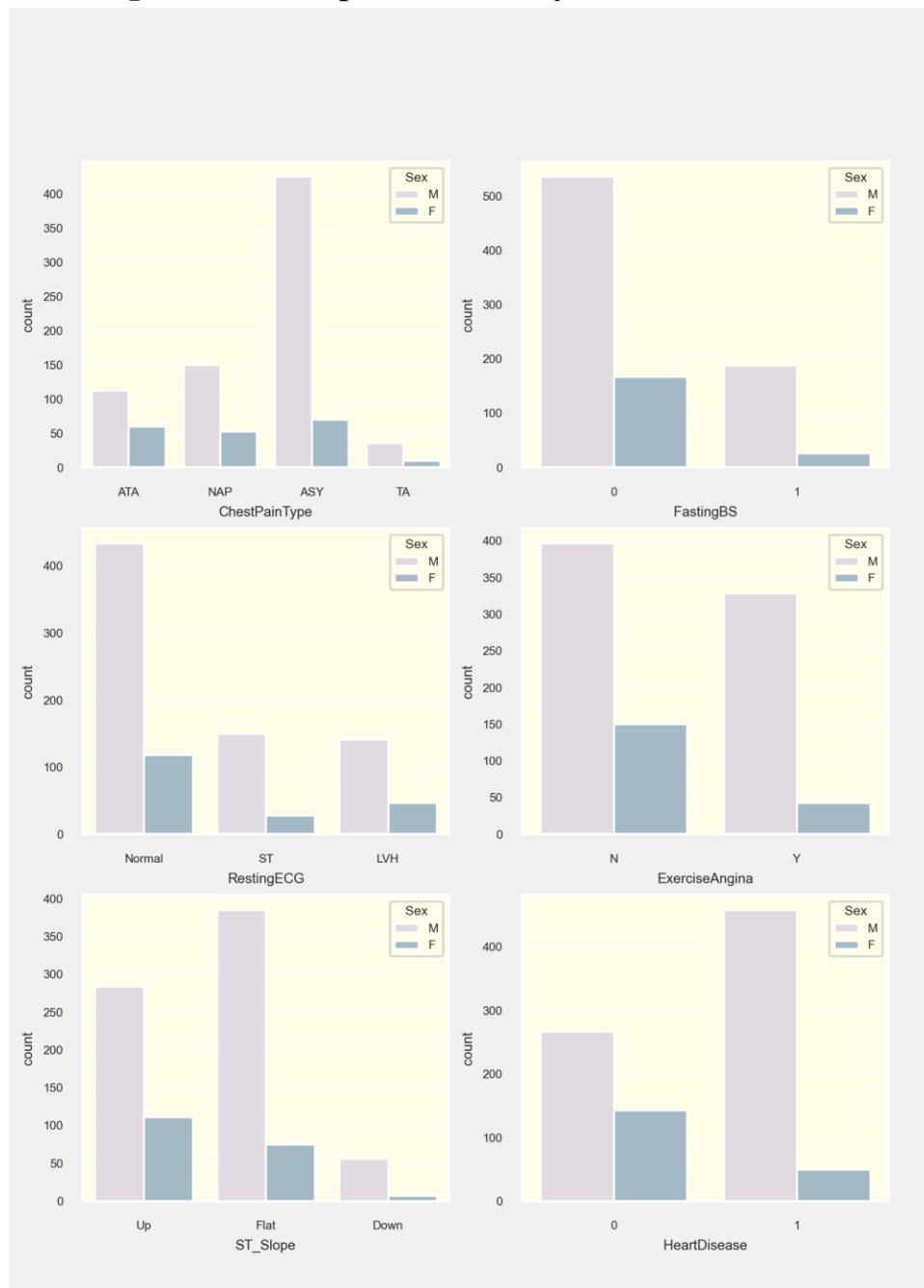
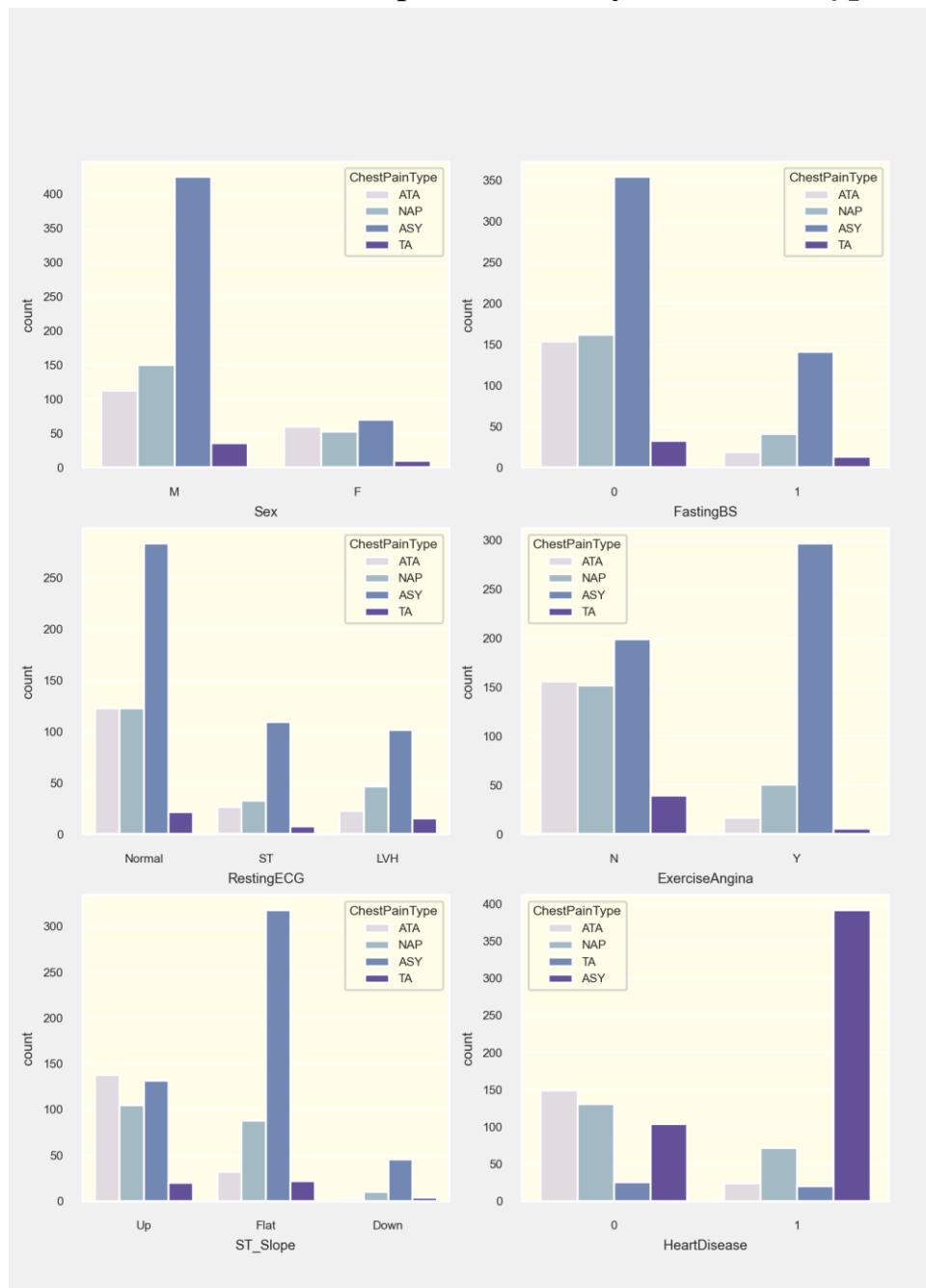






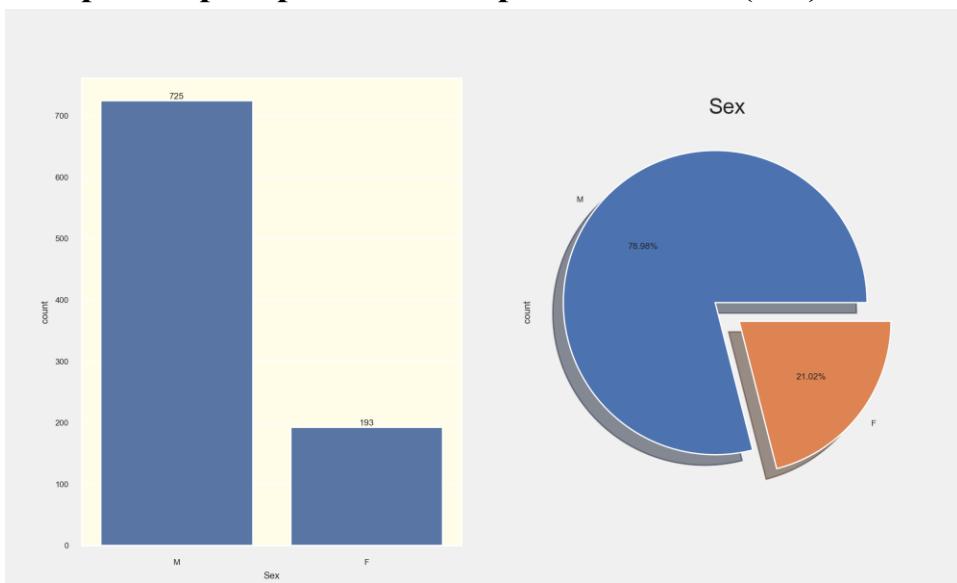
График №35. Столбиковая диаграмма распределения переменных: 'Sex', 'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST\_Slope', 'HeartDisease' по категориям и классу 'ChestPainType'







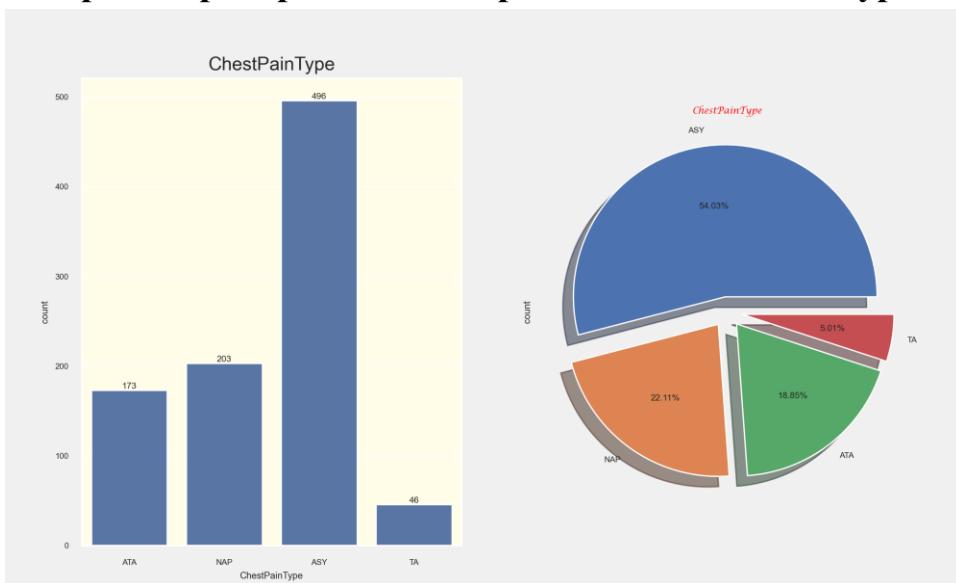
### График №36. Комбинированная (столбиковая и секторная) диаграмма распределения переменной пол (Sex)







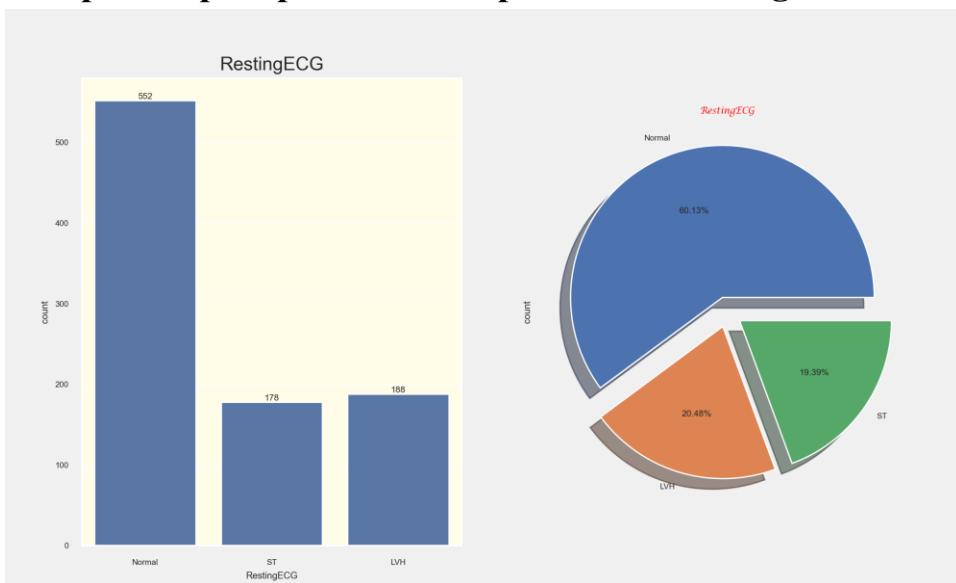
## График №37. Комбинированная (столбиковая и секторная) диаграмма распределения переменной ChestPainType







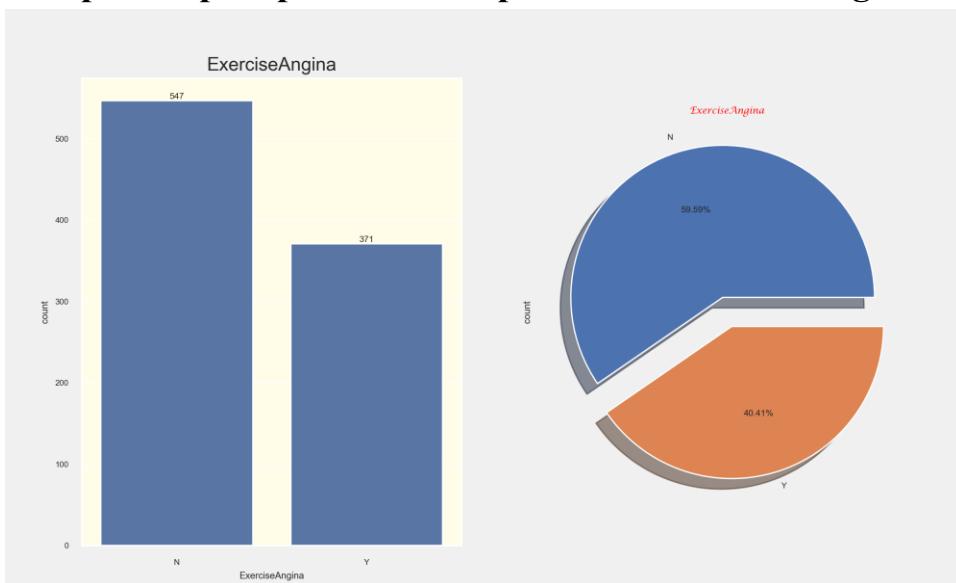
### График №38. Комбинированная (столбиковая и секторная) диаграмма распределения переменной RestingECG







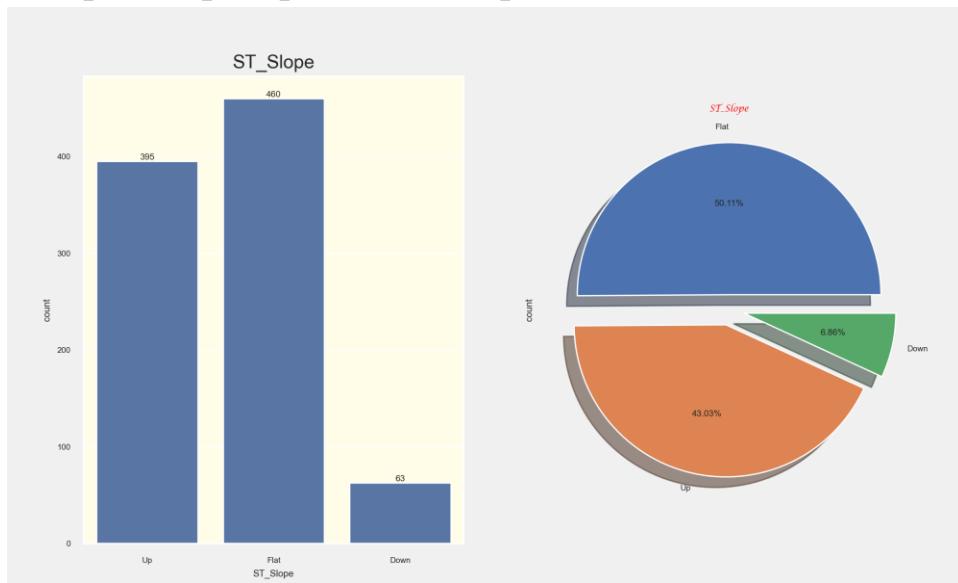
### График №39. Комбинированная (столбиковая и секторная) диаграмма распределения переменной ExerciseAngina







## График №40. Комбинированная (столбиковая и секторная) диаграмма распределения переменной ST\_Slope







## График №41. Комбинированная (столбиковая и секторная) диаграмма распределения переменной Cholesterol\_Category

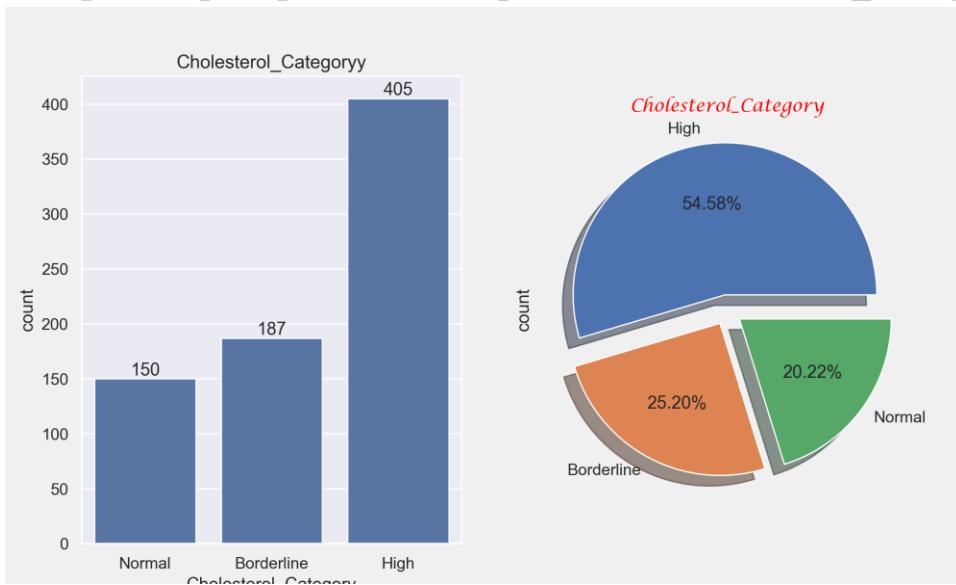
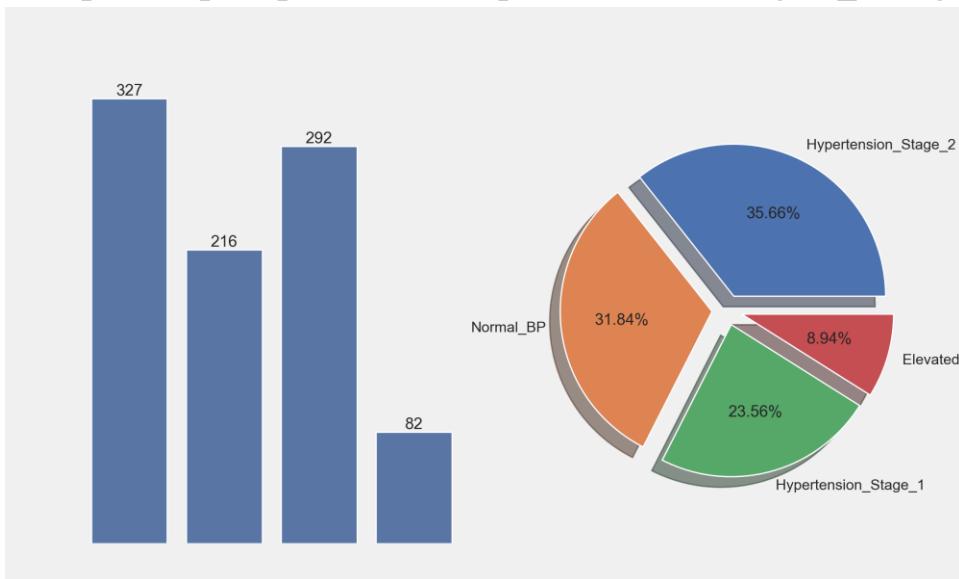






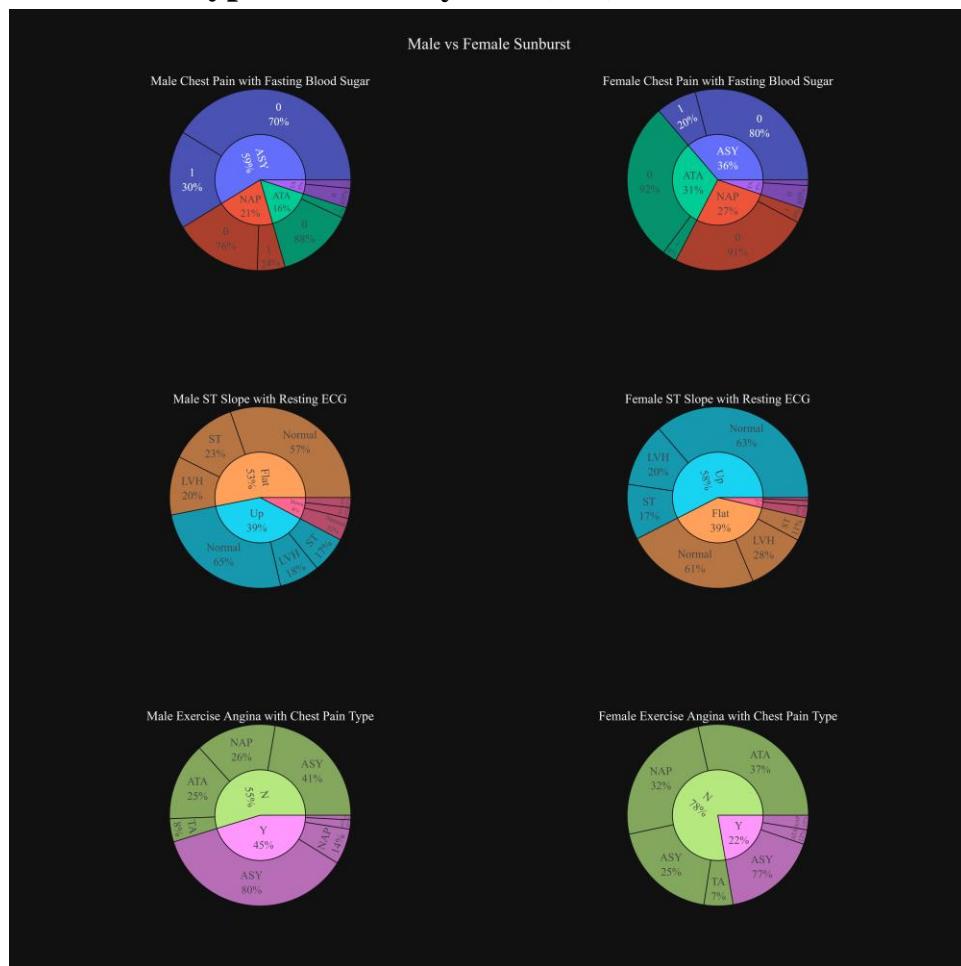
График №42. Комбинированная (столбиковая и секторная) диаграмма распределения переменной RestingBP\_Category







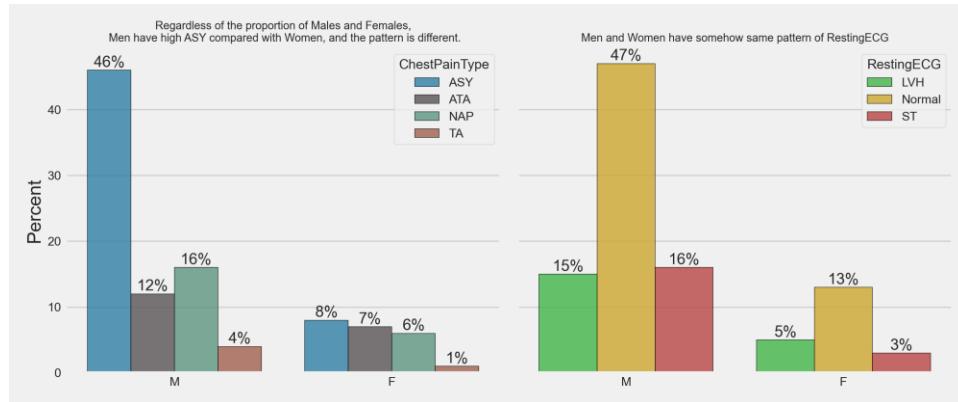
**График №43. Двойная секторная диаграмма (Sunburst, 6 субграфиков) распределения пар переменных: ['ChestPainType', 'FastingBS'], ['ST\_Slope', 'RestingECG'], ['ExerciseAngina', 'ChestPainType'] по классу пол (Sex)**







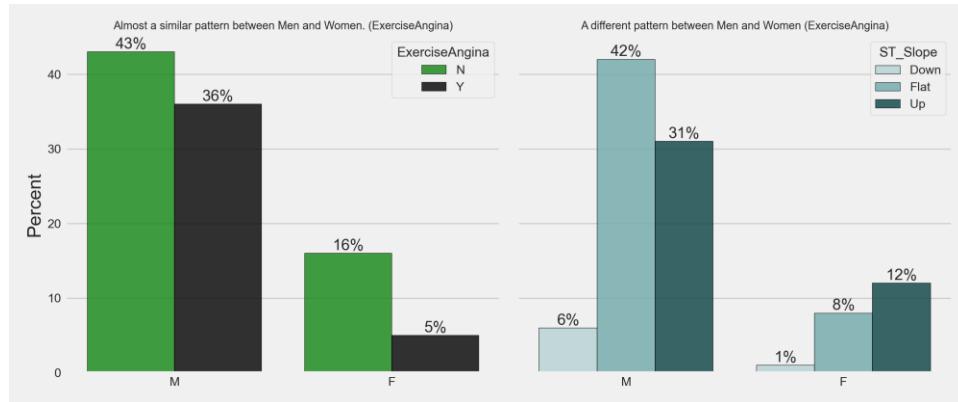
**График №44. Столбиковая диаграмма распределения переменных: RestingECG, ChestPainType (2 субграфика на одной панели) по классу пол (Sex)**







**График №45. Столбиковая диаграмма распределения переменных: ST\_Slope, ExerciseAngina (2 субграфика на одной панели) по классу пол (Sex)**







## Заключение по результатам EDA

EDA на наборе данных показал нам, как каждая переменная связана с переменной отклика и как мы можем сделать нашу модель эффективной, используя различные методы EDA. Визуализации данных поднимают наше понимание набора данных на более высокий уровень, позволяя нам делать выводы о направлении дальнейшего анализа.

Интеграция классификатора дерева принятия решений в наш анализ расширяет прогностические возможности нашей модели. В этом разделе мы не только изучили набор данных с помощью методов EDA, но и сделали еще один шаг вперед, внедрив модель машинного обучения. Такой целостный подход позволяет нам использовать сильные стороны как статистического анализа, так и прогнозного моделирования, способствуя более глубокому пониманию сложной динамики, связанной с прогнозированием ССЗ.



## Моделирование и Анализ данных

Для проведения моделирования и анализа данных нами использованы 18 моделей Машинного обучения. Ниже приведены выходные результаты, полученные в результате работы каждой модели.

Результатами являются:

- Таблица классификации
- Confusion Matrix
- ROC Curve
- Score plot

### Модель: Linear Regression

Linear Regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. Reference Wikipedia.

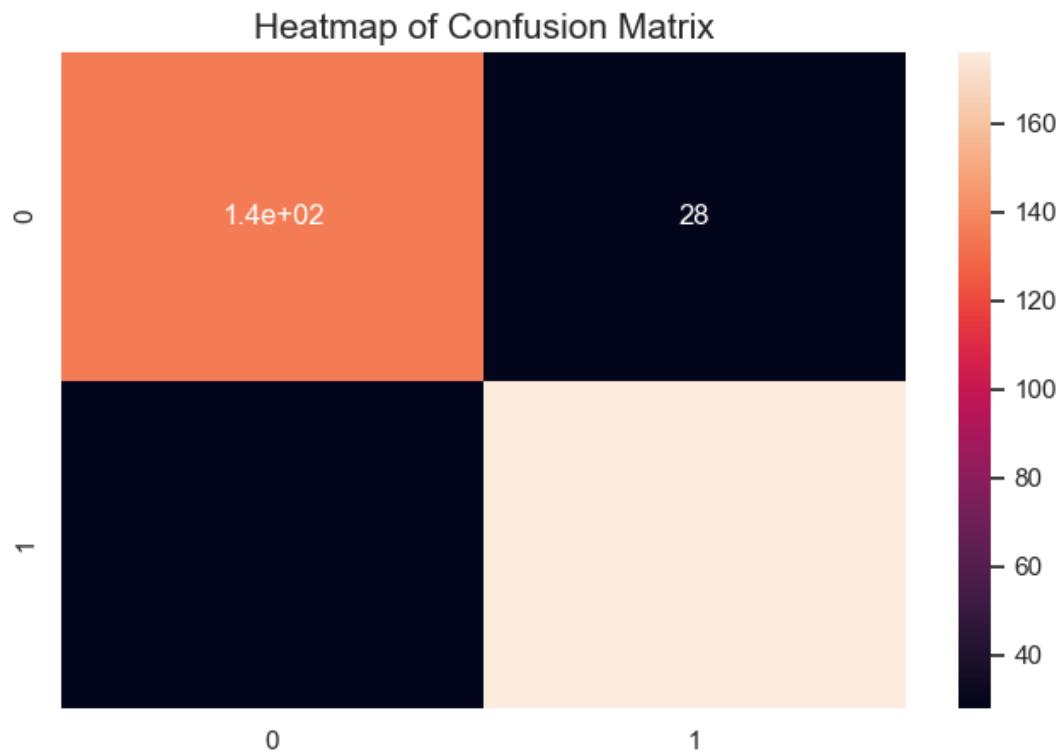
Таблица №15. Таблица классификации

<i>Classes+Metrics</i>	<i>precision</i>	<i>recall</i>	<i>f1-score</i>	<i>support</i>
class 0	0.822	0.817	0.82	164.0
class 1	0.853	0.857	0.855	203.0
accuracy	0.839	0.839	0.839	0.839
macro avg	0.838	0.837	0.837	367.0
weighted avg	0.839	0.839	0.839	367.0

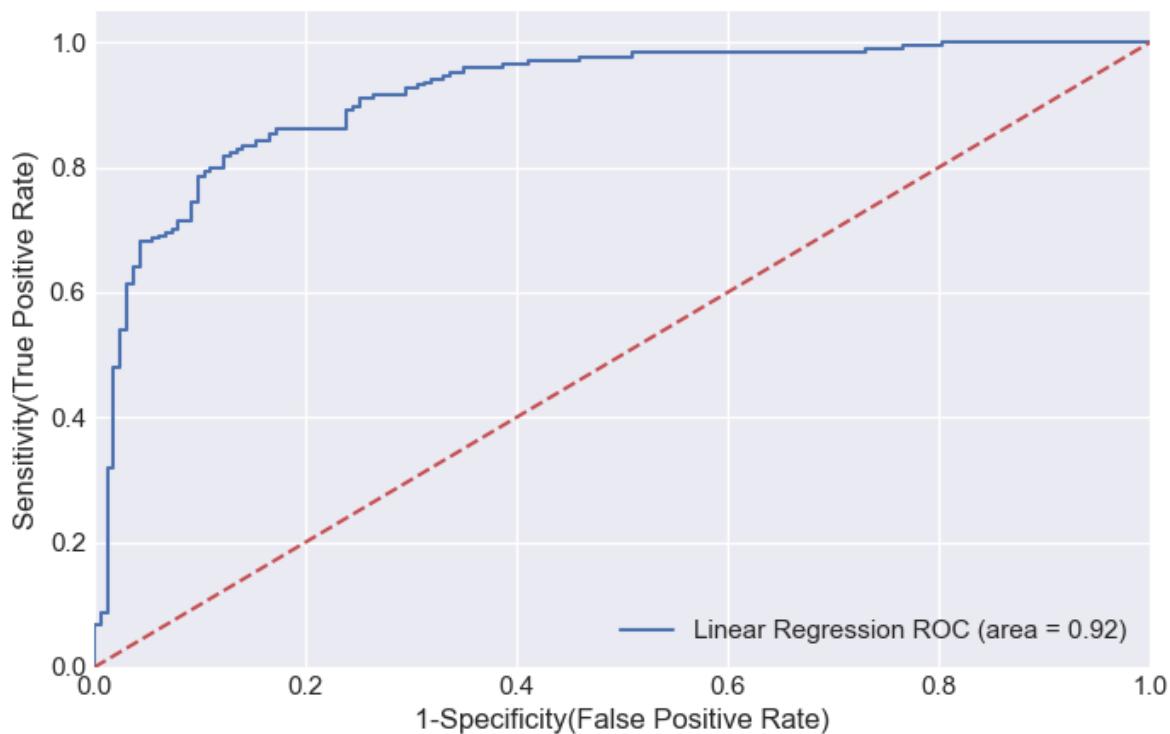
© Dr. Alexander Wagner. Все права охраняются законом



## График №46. Confusion Matrix

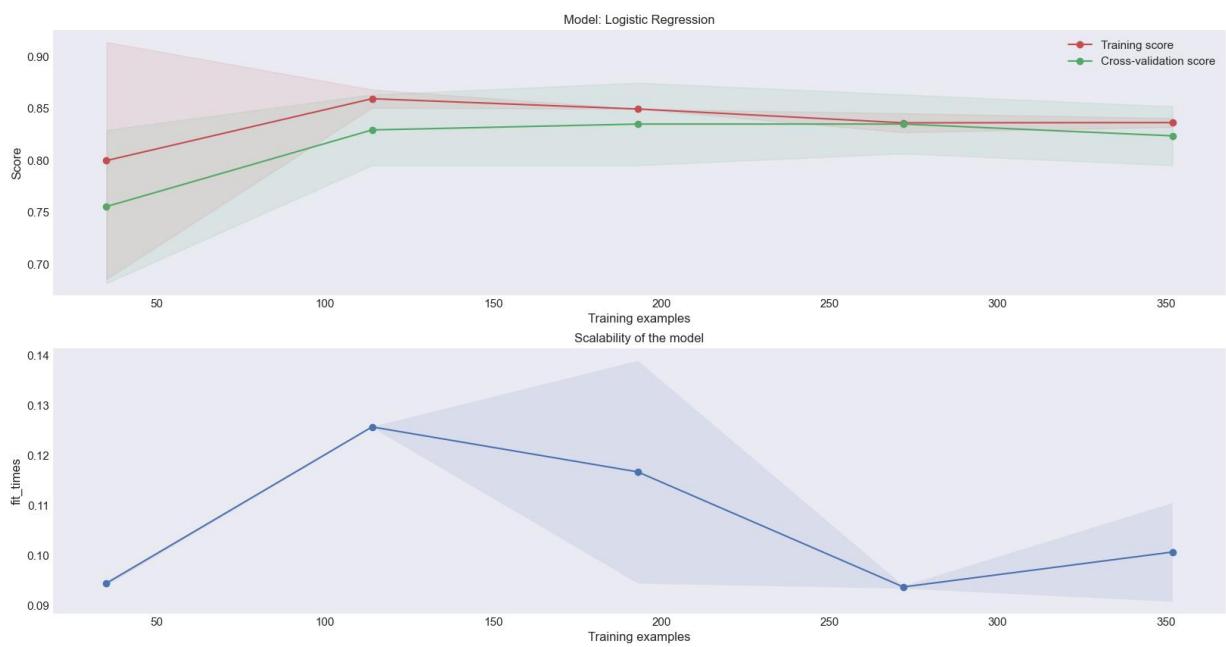


## График №47. ROC Curve





## График №48. Score plot



## Модель: Logistic Regression

Logistic Regression is a useful model to run early in the workflow. Logistic regression measures the relationship between the categorical dependent variable (feature) and one or more independent variables (features) by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Reference Wikipedia.

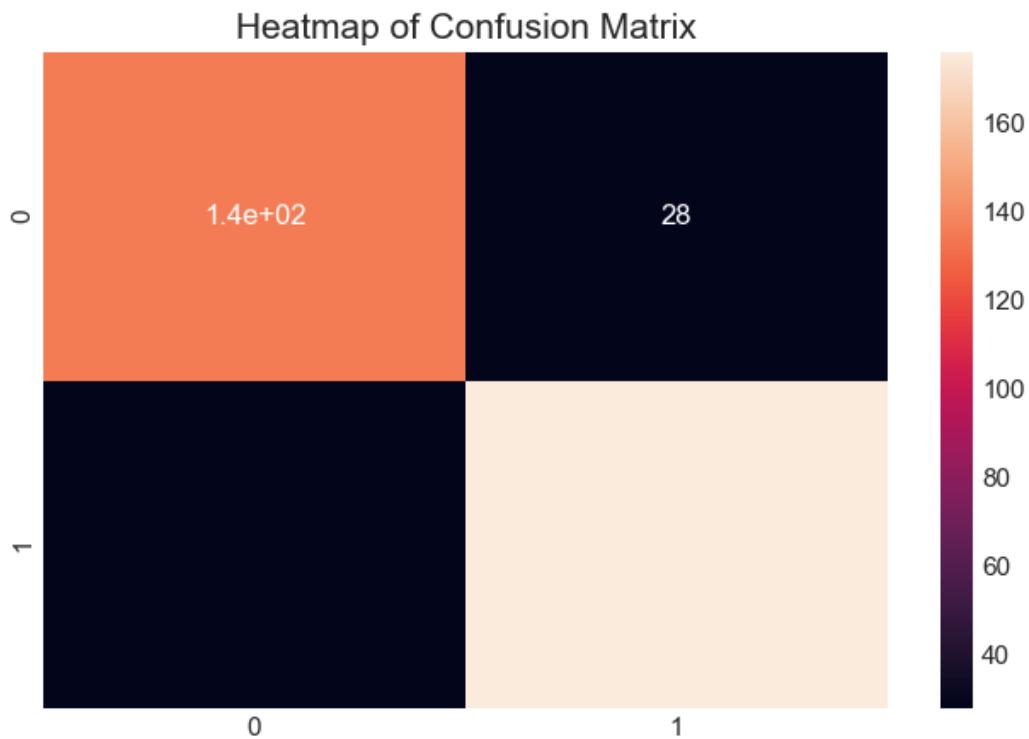
## Таблица №16. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.822	0.817	0.82	164.0
class 1	0.853	0.857	0.855	203.0
accuracy	0.839	0.839	0.839	0.839
macro avg	0.838	0.837	0.837	367.0
weighted avg	0.839	0.839	0.839	367.0

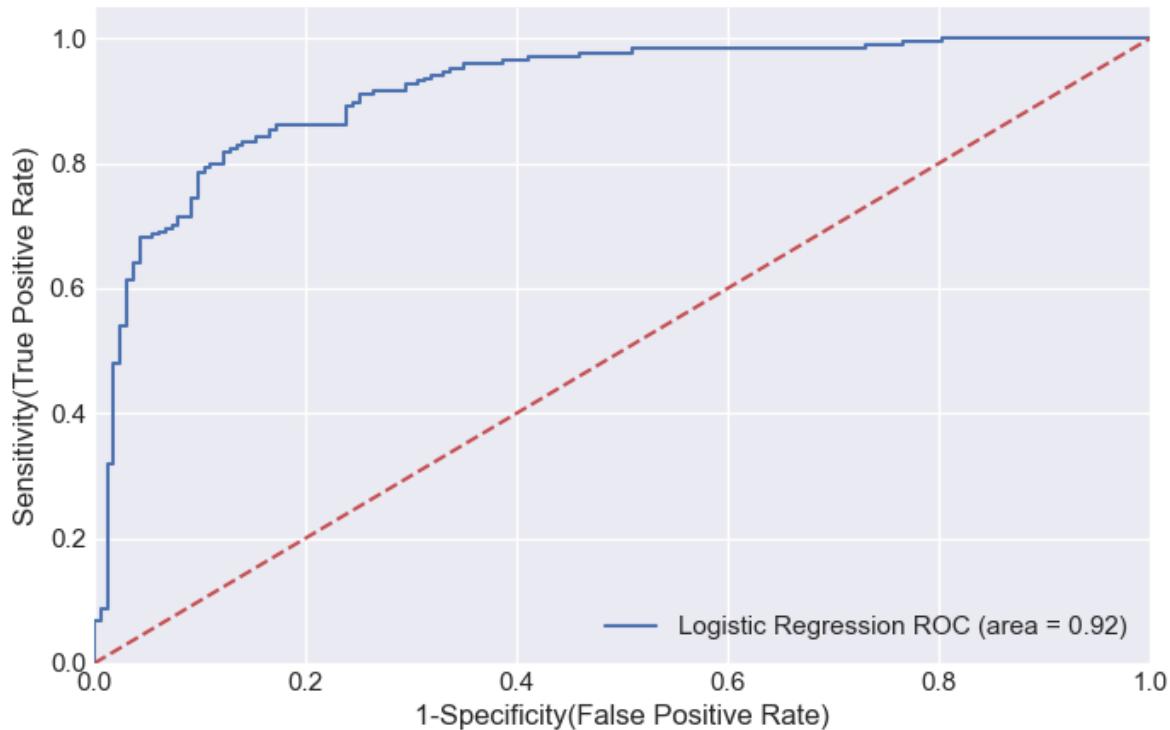
© Dr. Alexander Wagner. Все права охраняются законом



## График №49. Confusion Matrix

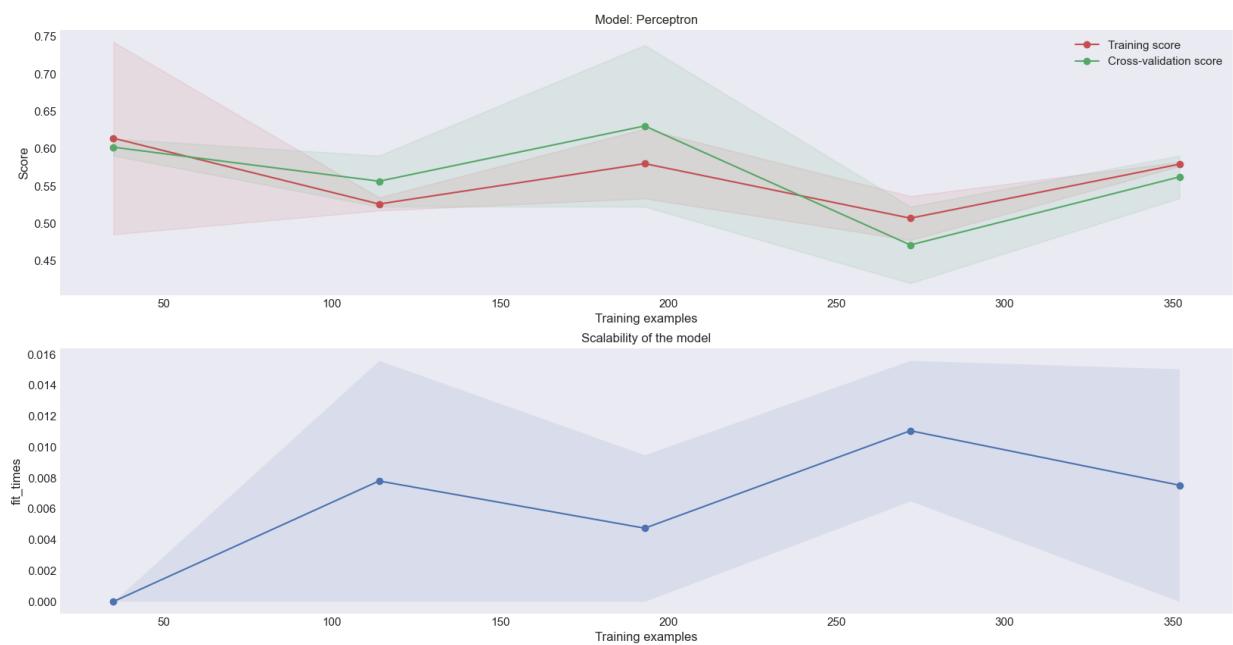


## График №50. ROC Curve





## График №51. Score plot



## Модель: Perceptron

Thanks to <https://www.kaggle.com/startupsci/titanic-data-science-solutions> The Perceptron is an algorithm for supervised learning of binary classifiers (functions that can decide whether an input, represented by a vector of numbers, belongs to some specific class or not). It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time. Reference Wikipedia.

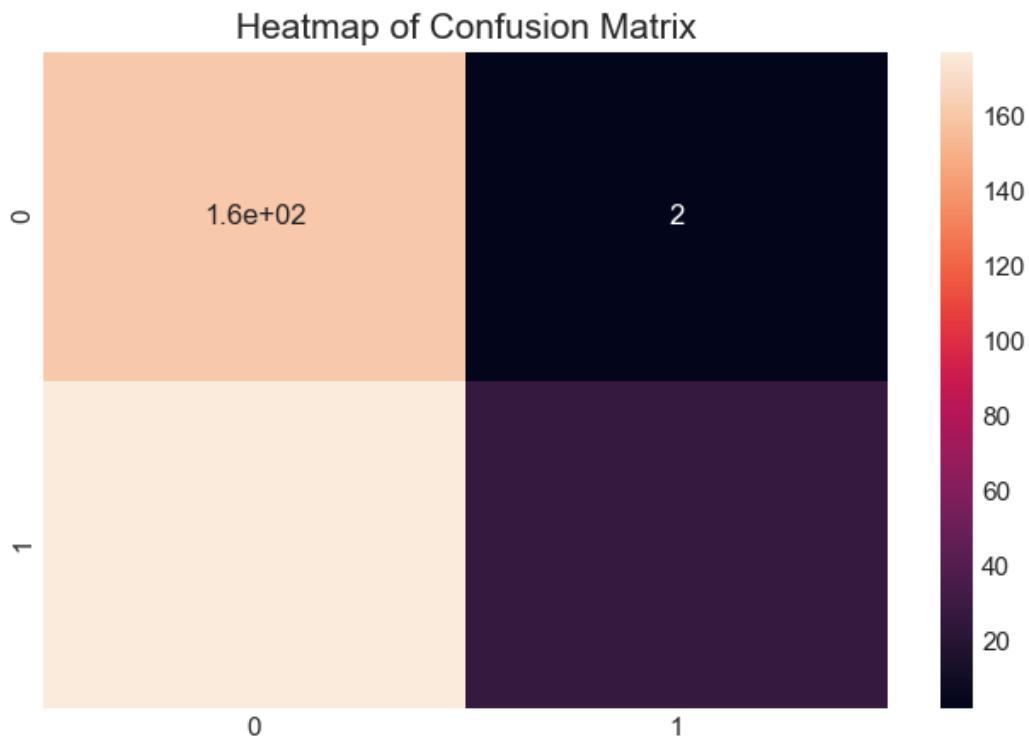
## Таблица №17. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.2	0.018	0.034	164.0
class 1	0.543	0.941	0.688	203.0
accuracy	0.529	0.529	0.529	0.529
macro avg	0.371	0.48	0.361	367.0
weighted avg	0.39	0.529	0.396	367.0

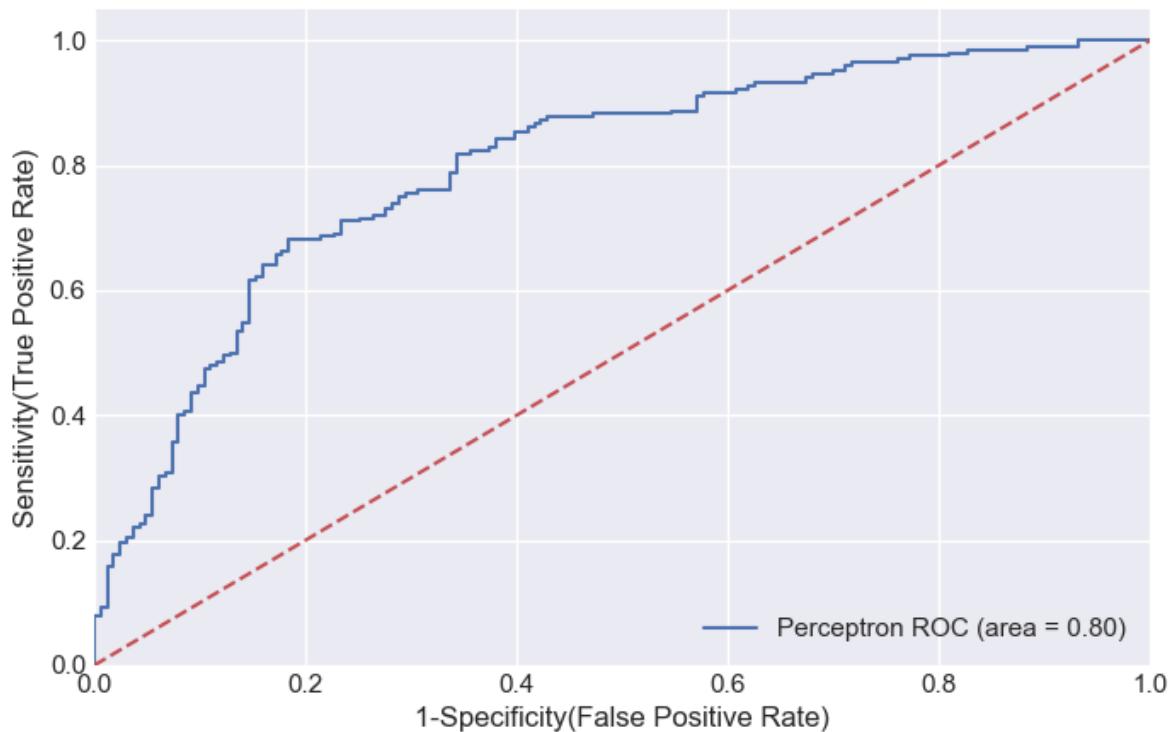
© Dr. Alexander Wagner. Все права охраняются законом



## График №52. Confusion Matrix

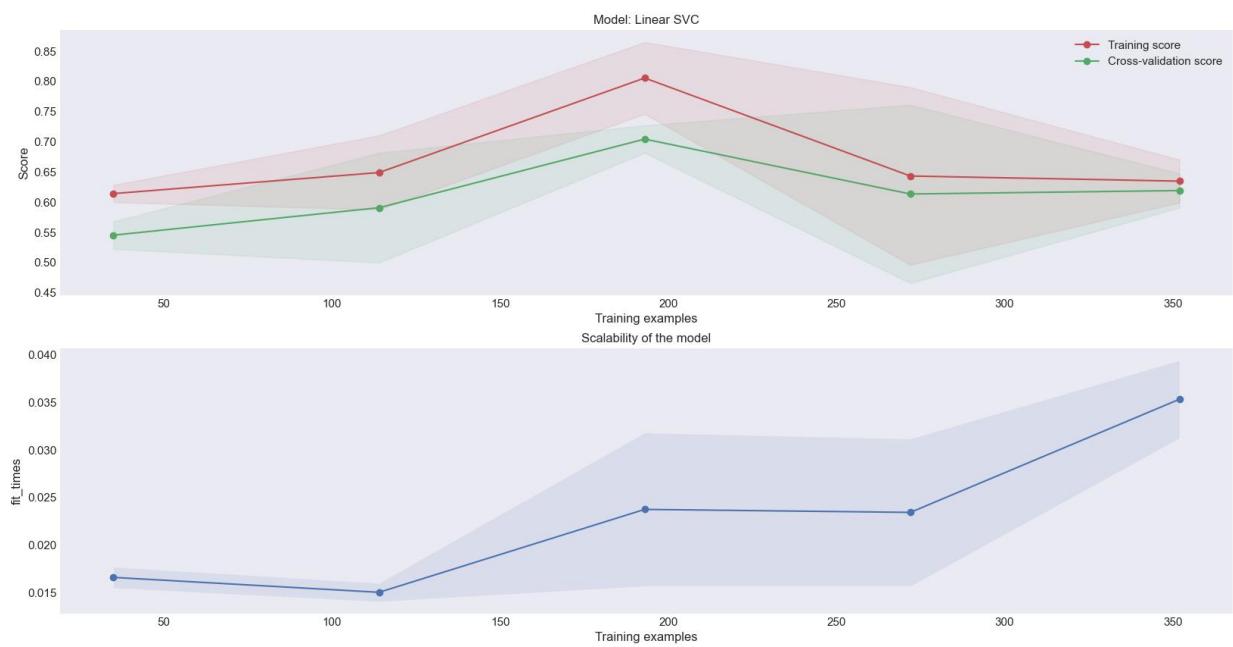


## График №53. ROC Curve





## График №54. Score plot



### Модель: Linear SVC

Linear SVC is a similar to SVM method. Its also builds on kernel functions but is appropriate for unsupervised learning. Reference Wikipedia.

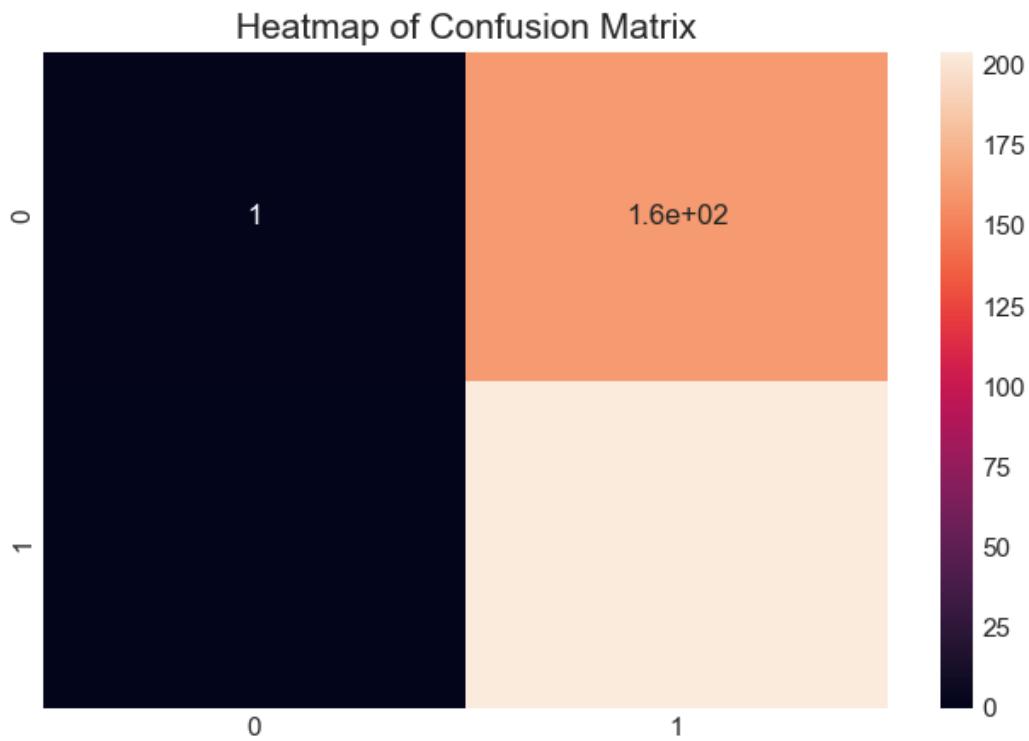
Таблица №18. Таблица классификации

<i>Classes+Metrics</i>	<i>precision</i>	<i>recall</i>	<i>f1-score</i>	<i>support</i>
class 0	0.822	0.817	0.82	164.0
class 1	0.853	0.857	0.855	203.0
accuracy	0.839	0.839	0.839	0.839
macro avg	0.838	0.837	0.837	367.0
weighted avg	0.839	0.839	0.839	367.0

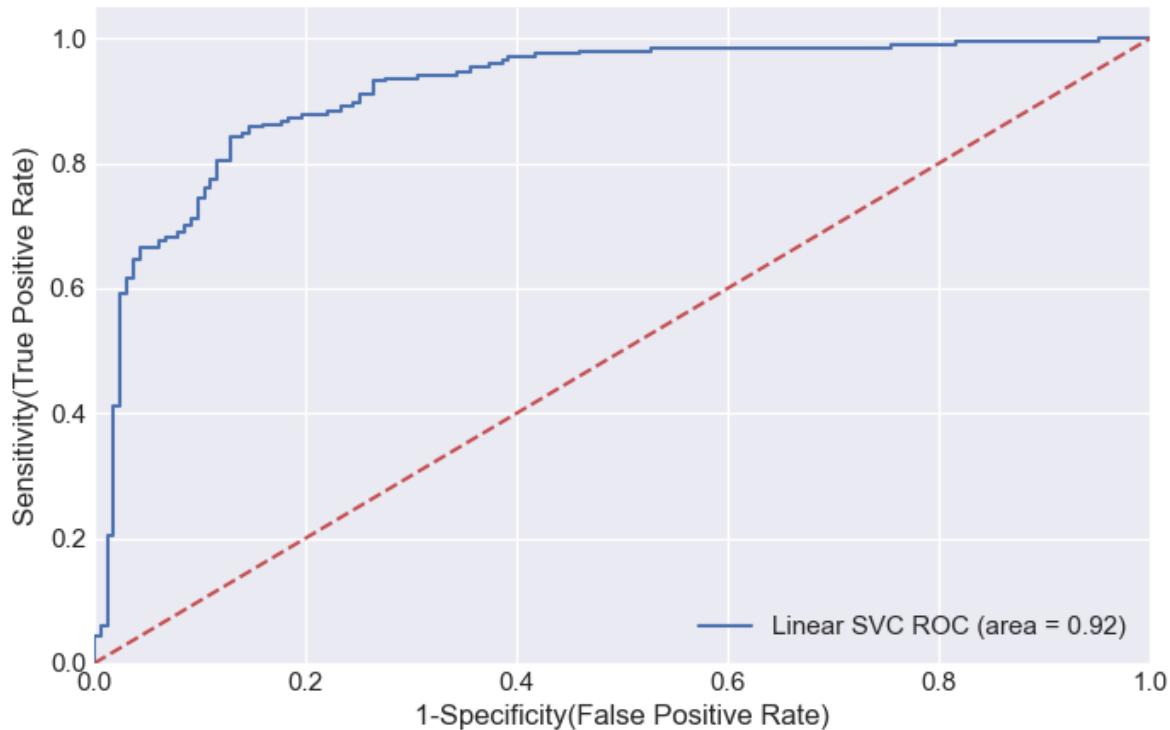
© Dr. Alexander Wagner. Все права охраняются законом



## График №55. Confusion Matrix

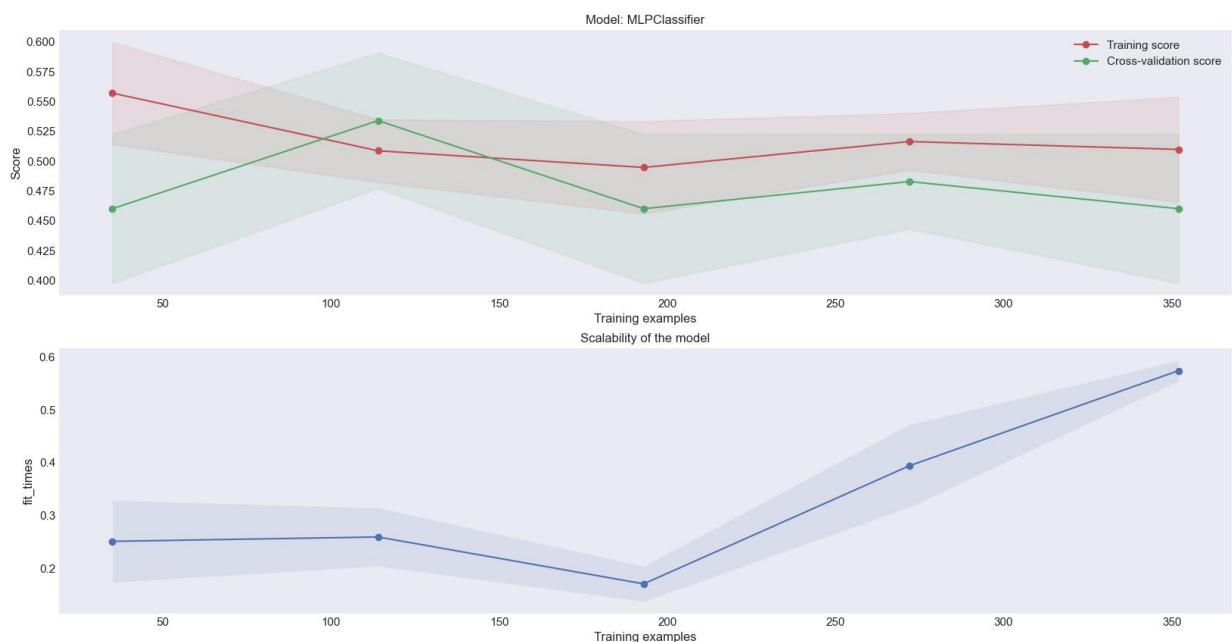


## График №56. ROC Curve





## График №57. Score plot



## Модель: MLPClassifier

The MLPClassifier optimizes the squared-loss using LBFGS or stochastic gradient descent by the Multi-layer Perceptron regressor. Reference Sklearn documentation.

Thanks to: [https://scikit-learn.org/stable/modules/generated/sklearn.neural\\_network.MLPRegressor.html#sklearn.neural\\_network.MLPRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.neural_network.MLPRegressor) <https://stackoverflow.com/questions/44803596/scikit-learn-mlpregressor-performance-cap>

Linear SVC is a similar to SVM method. Its also builds on kernel functions but is appropriate for unsupervised learning. Reference Wikipedia.

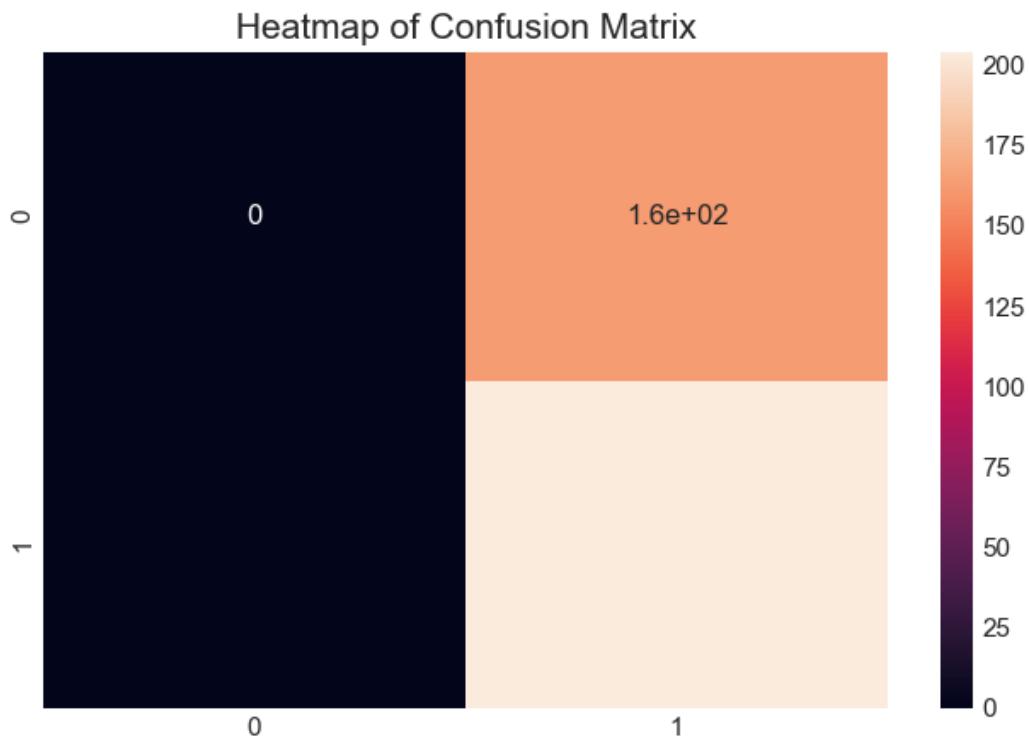
## Таблица №19. Таблица классификации

<i>Classes+Metrics</i>	<i>precision</i>	<i>recall</i>	<i>f1-score</i>	<i>support</i>
class 0	0.0	0.0	0.0	164.0
class 1	0.552	0.995	0.71	203.0
accuracy	0.55	0.55	0.55	0.55
macro avg	0.276	0.498	0.355	367.0
weighted avg	0.305	0.55	0.393	367.0

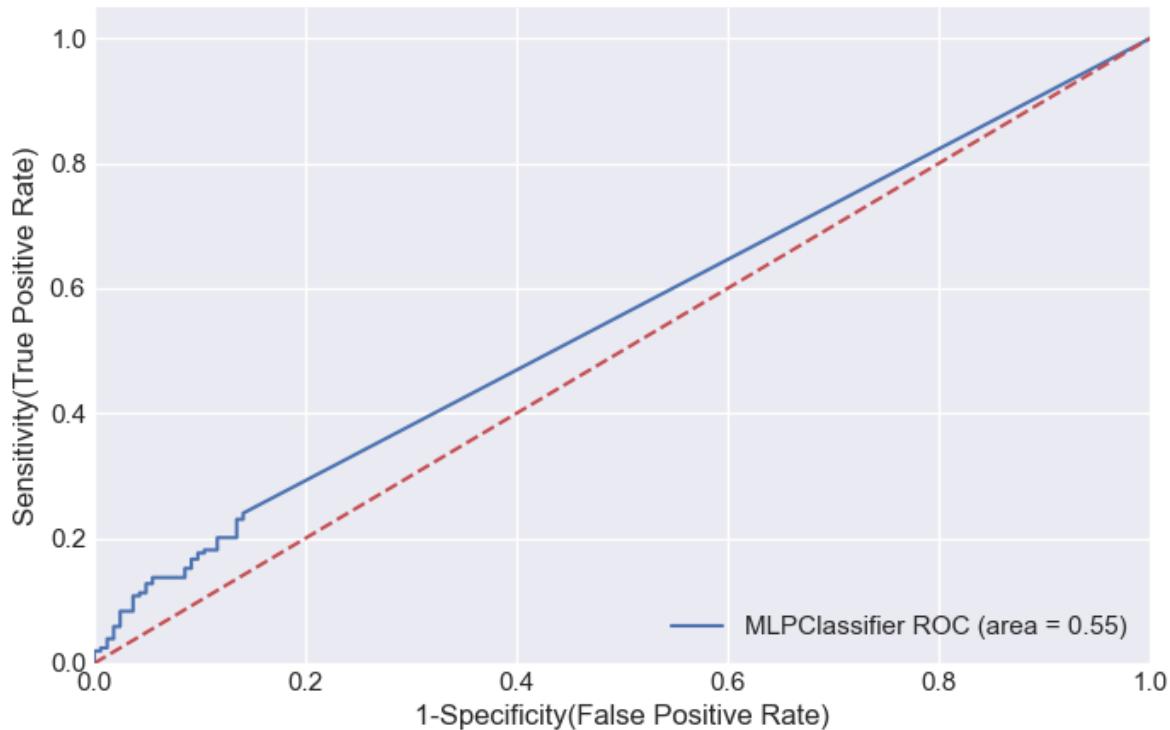
© Dr. Alexander Wagner. Все права охраняются законом



## График №58. Confusion Matrix

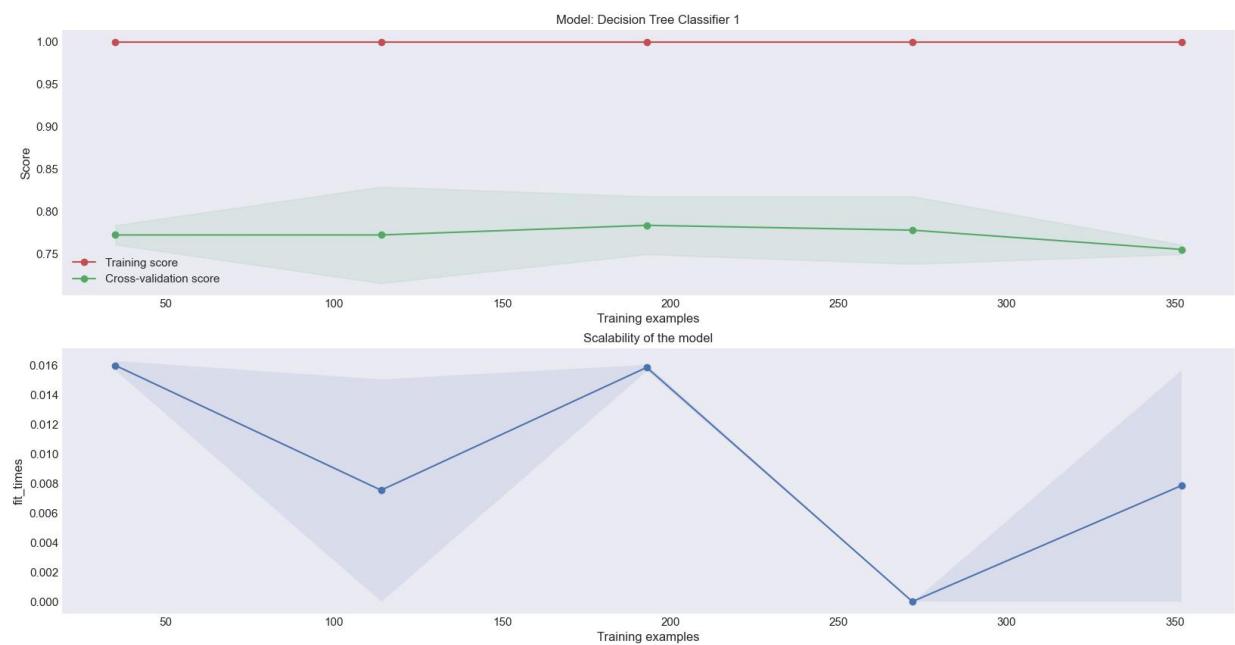


## График №59. ROC Curve





## График №60. Score plot



## Модель: Decision Tree Classifier 1

This model uses a Decision Tree as a predictive model which maps features (tree branches) to conclusions about the target value (tree leaves). Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Reference Wikipedia.

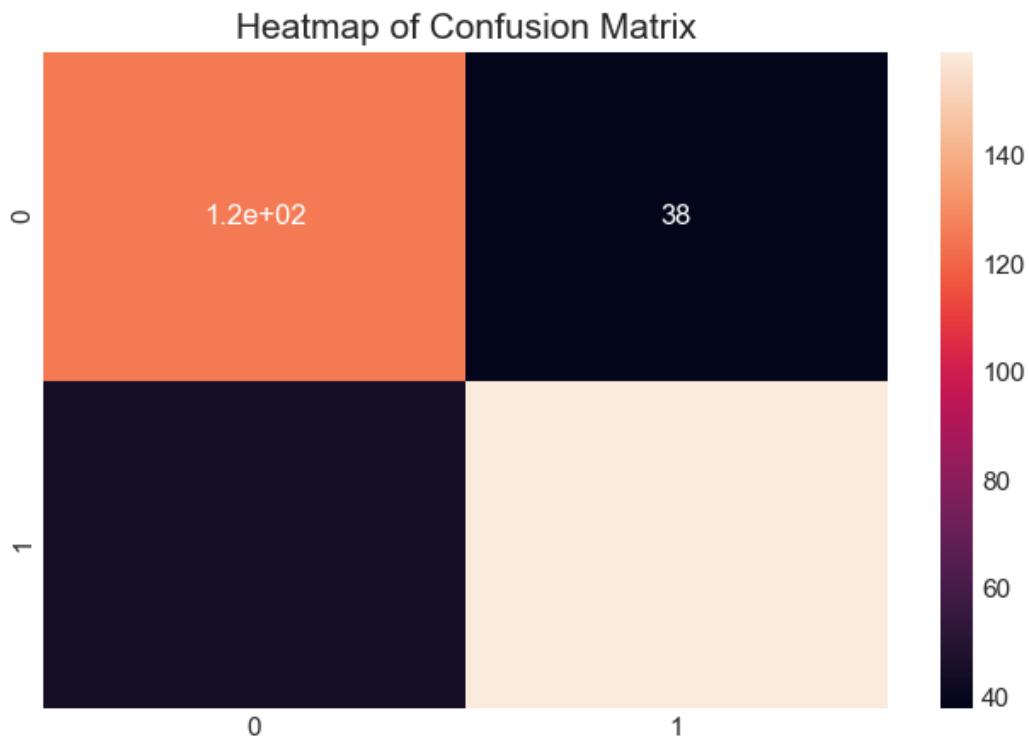
## Таблица №20. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.756	0.774	0.765	164.0
class 1	0.814	0.798	0.806	203.0
accuracy	0.787	0.787	0.787	0.787
macro avg	0.785	0.786	0.786	367.0
weighted avg	0.788	0.787	0.788	367.0

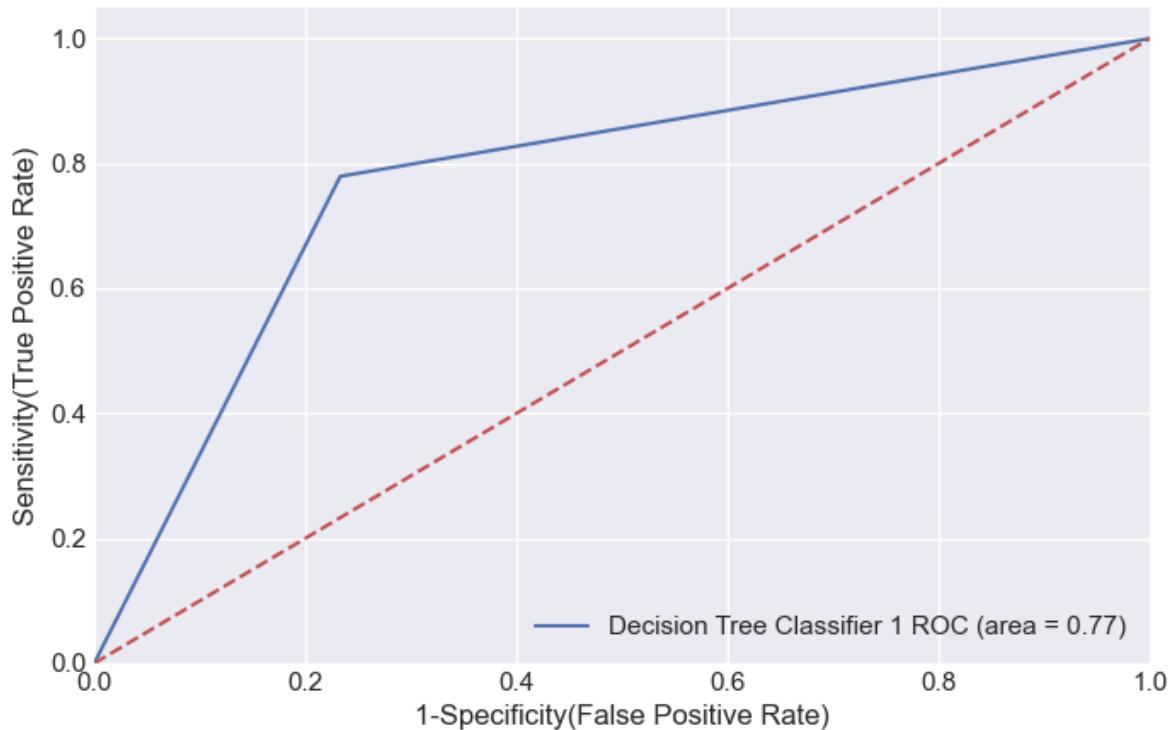
© Dr. Alexander Wagner. Все права охраняются законом



## График №61. Confusion Matrix

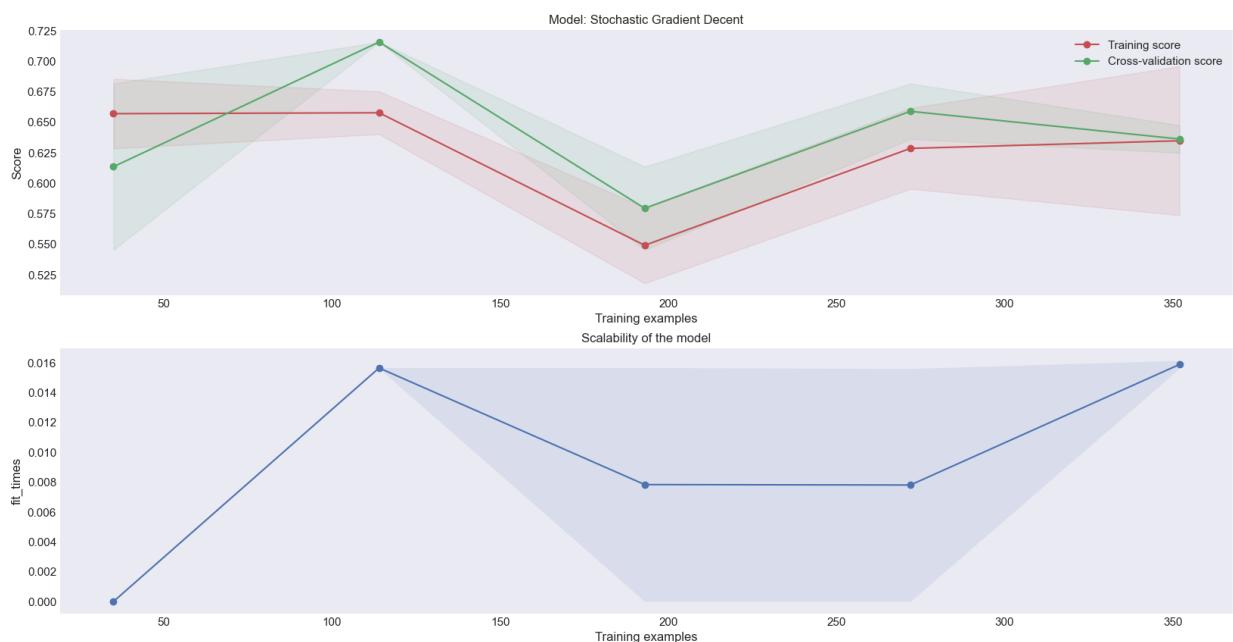


## График №62. ROC Curve





## График №63. Score plot



## Модель: Stochastic Gradient Decent

Stochastic gradient descent (often abbreviated SGD) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable or subdifferentiable). It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate thereof (calculated from a randomly selected subset of the data). Especially in big data applications this reduces the computational burden, achieving faster iterations in trade for a slightly lower convergence rate. Reference Wikipedia.

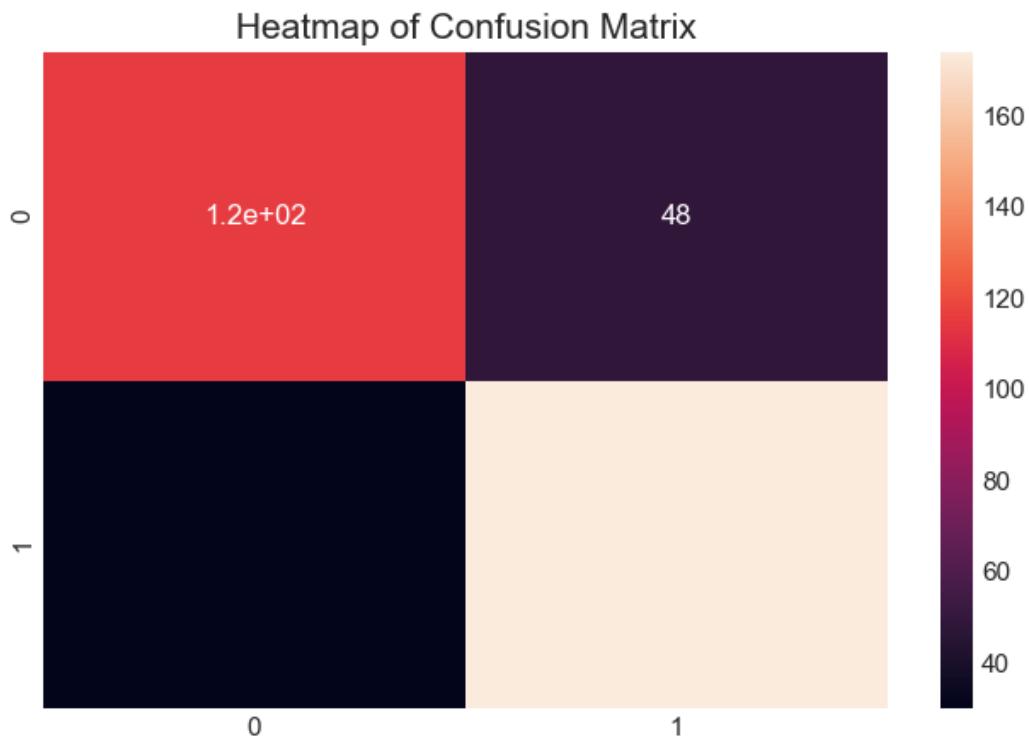
Таблица №21. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.63	0.841	0.721	164.0
class 1	0.824	0.601	0.695	203.0
accuracy	0.708	0.708	0.708	0.708
macro avg	0.727	0.721	0.708	367.0
weighted avg	0.738	0.708	0.707	367.0

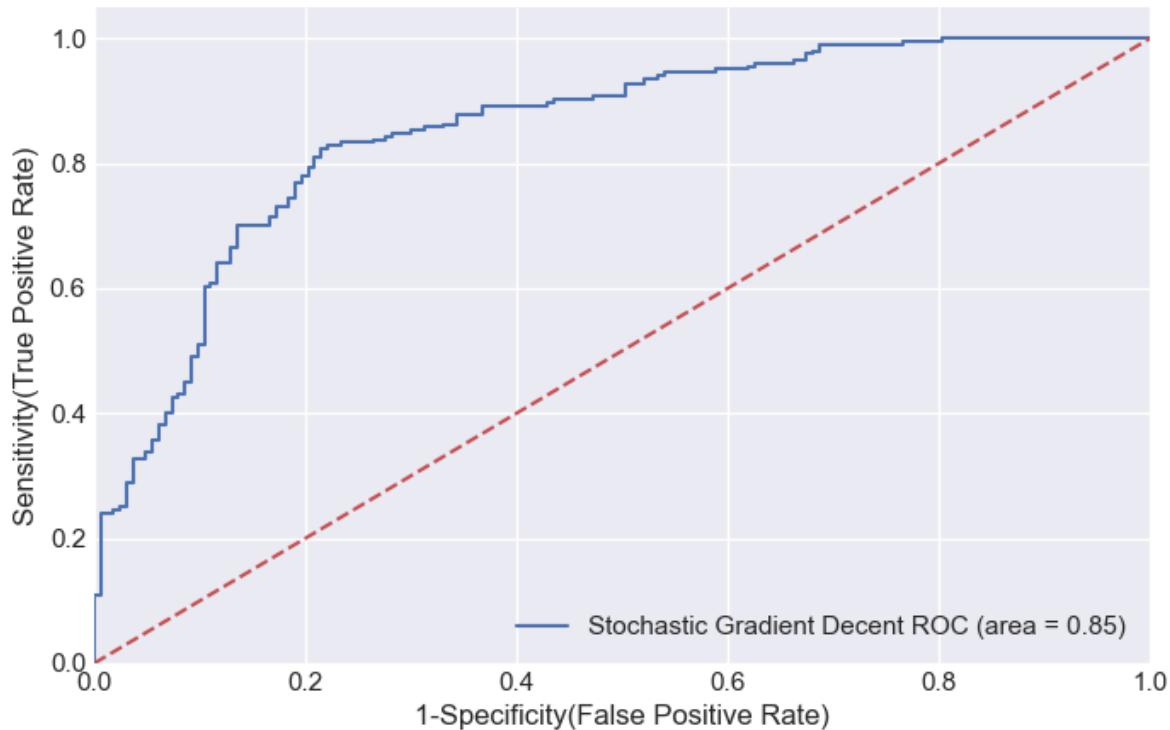
© Dr. Alexander Wagner. Все права охраняются законом



## График №64. Confusion Matrix

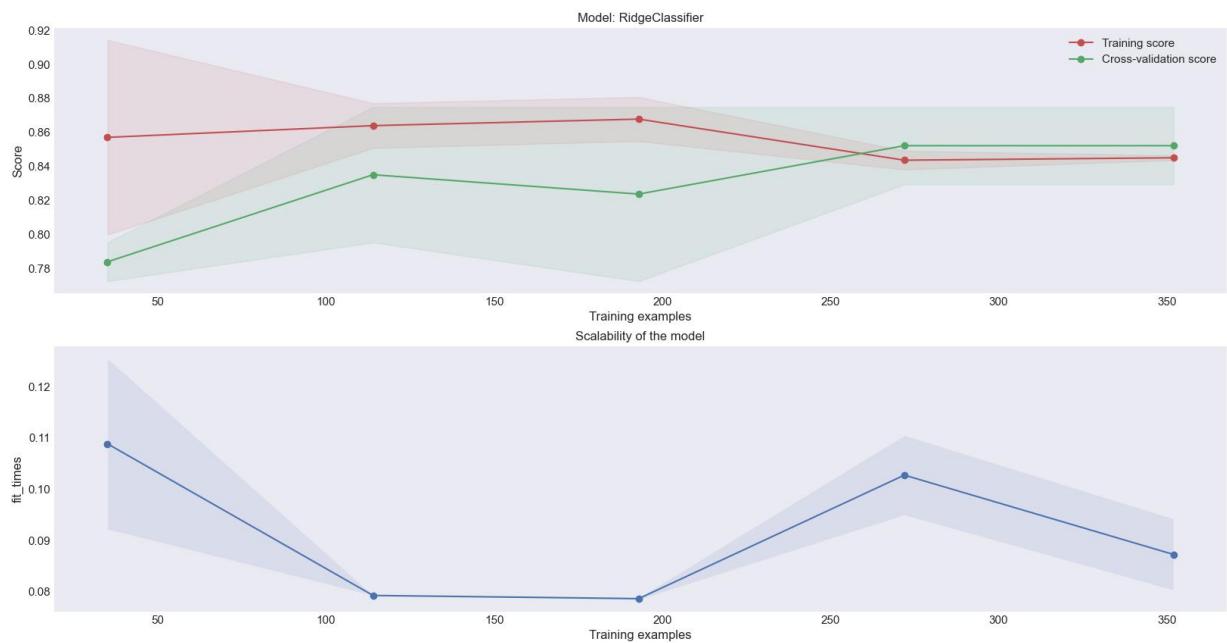


## График №65. ROC Curve





## График №66. Score plot



### Модель: RidgeClassifier

Tikhonov Regularization, colloquially known as Ridge Classifier, is the most commonly used regression algorithm to approximate an answer for an equation with no unique solution. This type of problem is very common in machine learning tasks, where the "best" solution must be chosen using limited data. If a unique solution exists, algorithm will return the optimal value. However, if multiple solutions exist, it may choose any of them. Reference Brilliant.org.

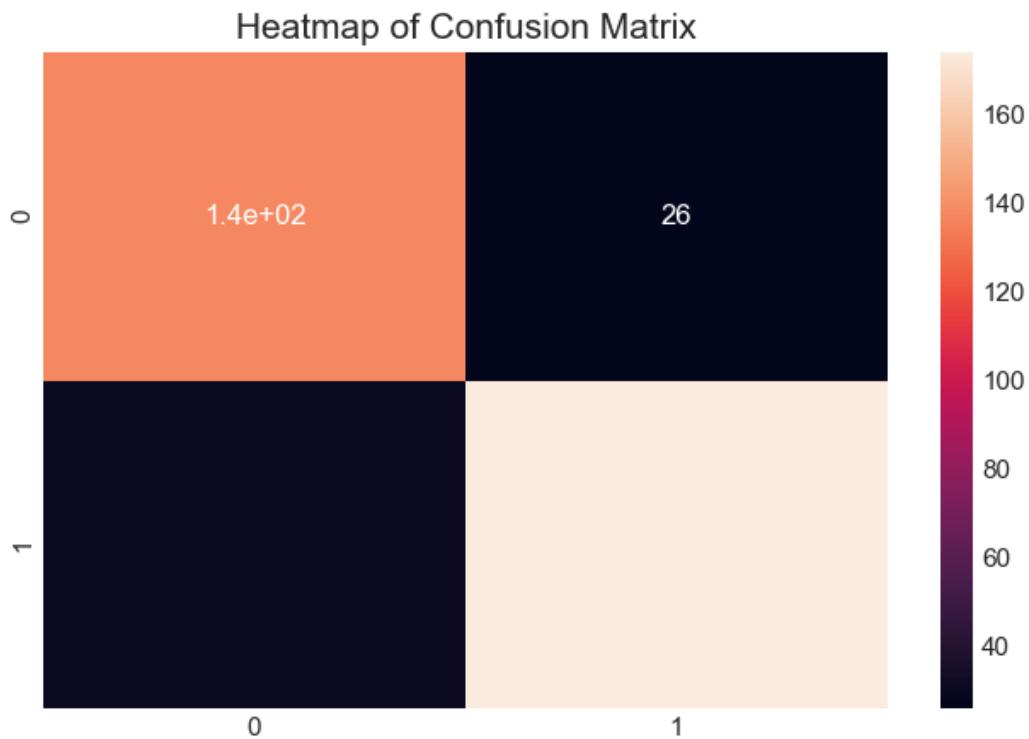
### Таблица №22. Таблица классификации

class 0	0.819	0.829	0.824	164.0
class 1	0.861	0.852	0.856	203.0
accuracy	0.842	0.842	0.842	0.842
macro avg	0.84	0.841	0.84	367.0
weighted avg	0.842	0.842	0.842	367.0

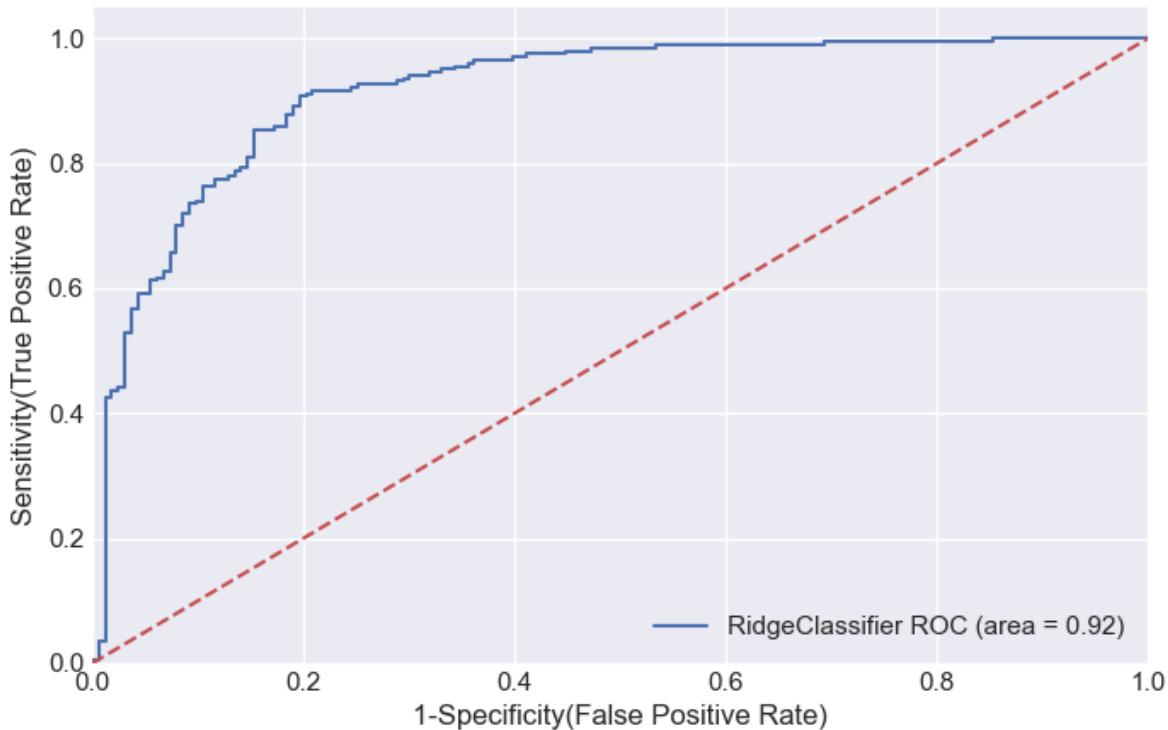
© Dr. Alexander Wagner. Все права охраняются законом



## График №67. Confusion Matrix

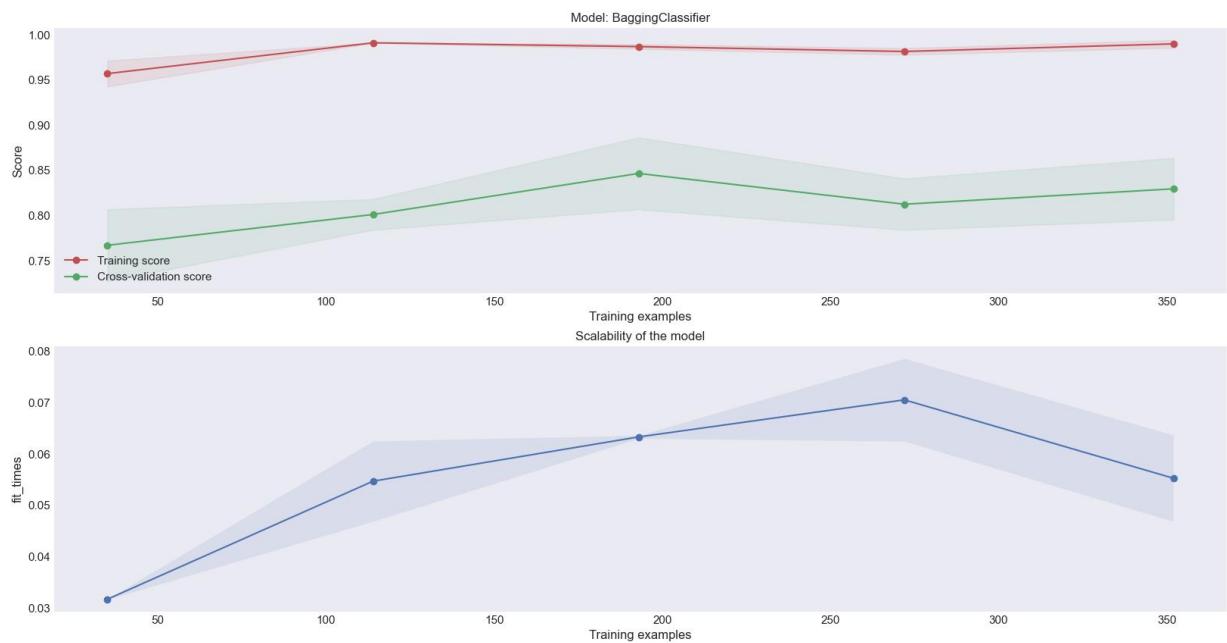


## График №68. ROC Curve





## График №69. Score plot



## Модель: BaggingClassifier

Bootstrap aggregating, also called Bagging, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting. Although it is usually applied to decision tree methods, it can be used with any type of method. Bagging is a special case of the model averaging approach. Bagging leads to "improvements for unstable procedures", which include, for example, artificial neural networks, classification and regression trees, and subset selection in linear regression. On the other hand, it can mildly degrade the performance of stable methods such as K-nearest neighbors. Reference Wikipedia.

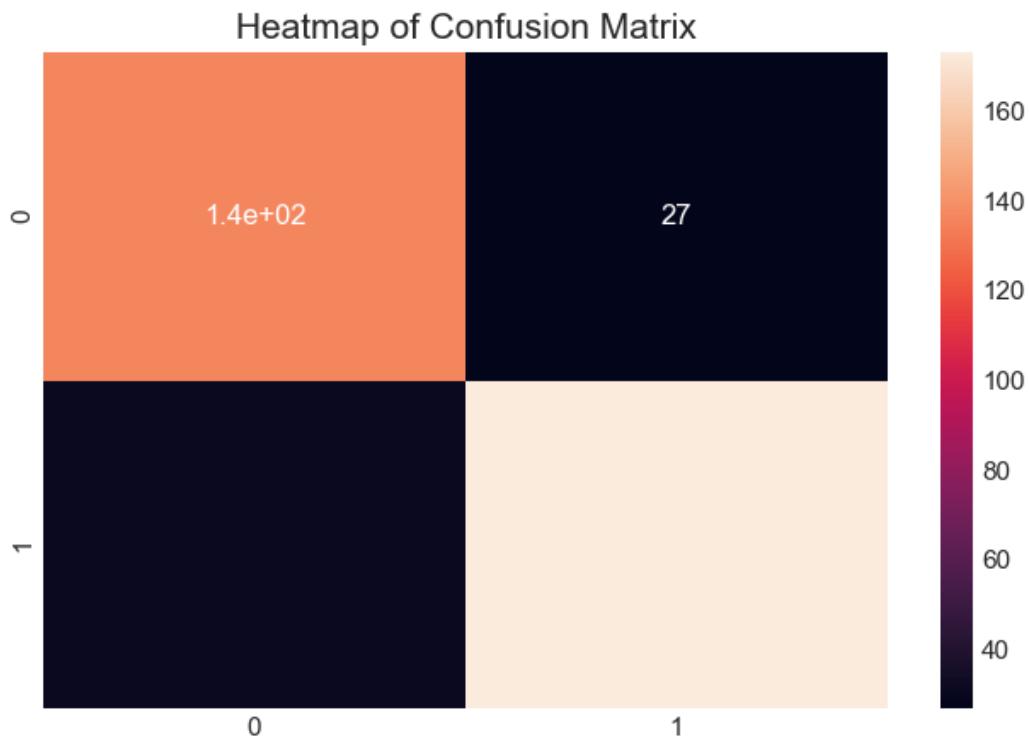
Таблица №23. Таблица классификации

<i>Classes+Metrics</i>	<i>precision</i>	<i>recall</i>	<i>f1-score</i>	<i>support</i>
class 0	0.799	0.799	0.799	164.0
class 1	0.837	0.837	0.837	203.0
accuracy	0.82	0.82	0.82	0.82
macro avg	0.818	0.818	0.818	367.0
weighted avg	0.82	0.82	0.82	367.0

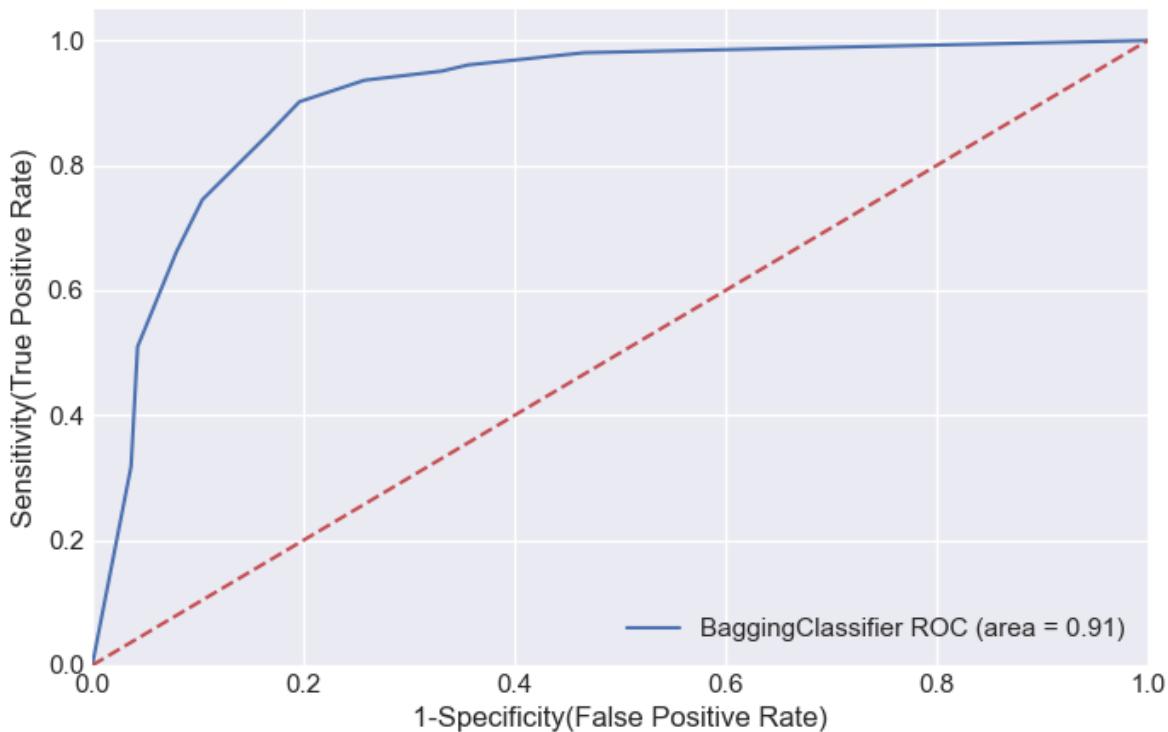
© Dr. Alexander Wagner. Все права охраняются законом



## График №70. Confusion Matrix

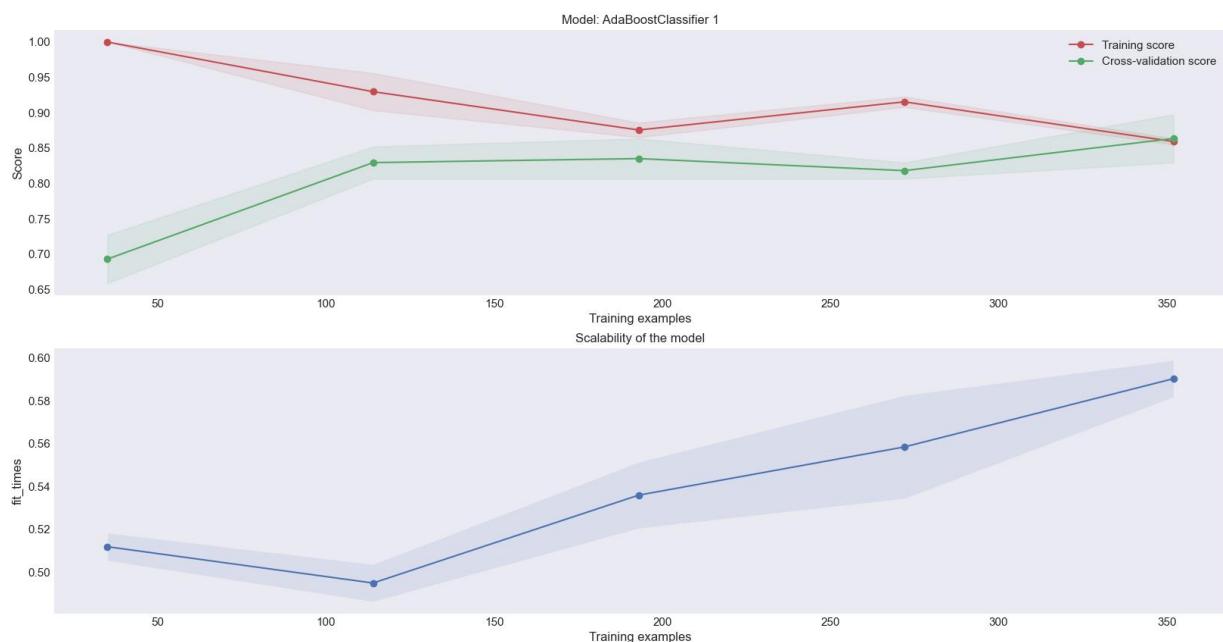


## График №71. ROC Curve





## График №72. Score plot



### Модель: AdaBoostClassifier 1

The core principle of AdaBoost ("Adaptive Boosting") is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying N weights to each of the training samples. Initially, those weights are all set to  $1/N$ , so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence. Reference sklearn documentation.

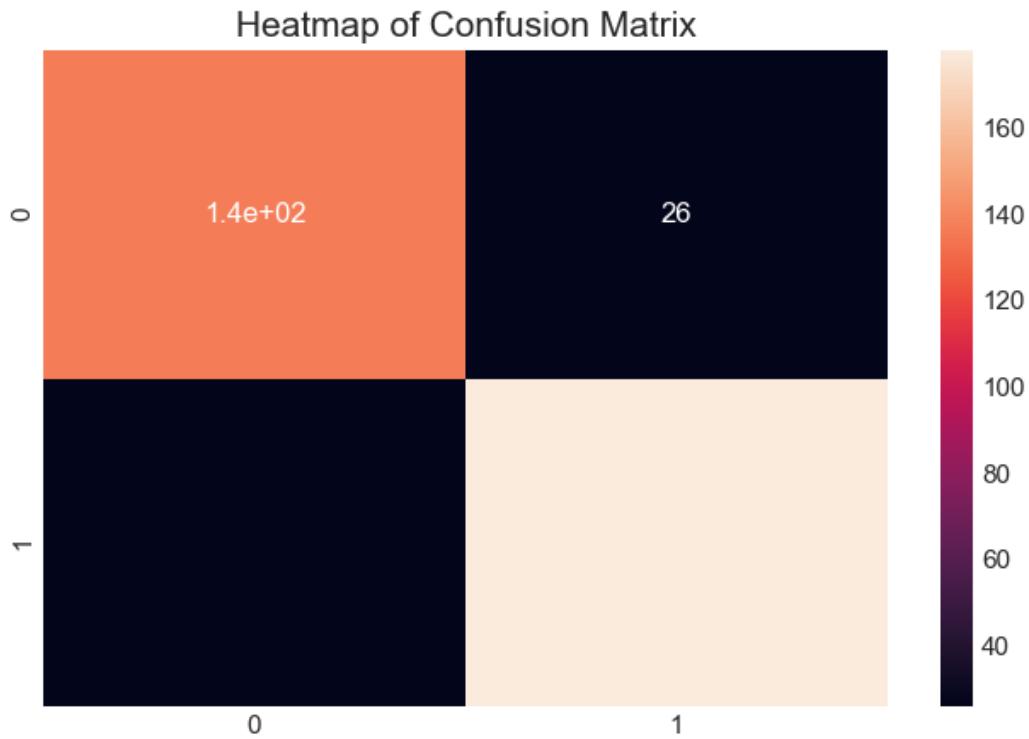
### Таблица №24. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.829	0.829	0.829	164.0
class 1	0.862	0.862	0.862	203.0
accuracy	0.847	0.847	0.847	0.847
macro avg	0.846	0.846	0.846	367.0
weighted avg	0.847	0.847	0.847	367.0

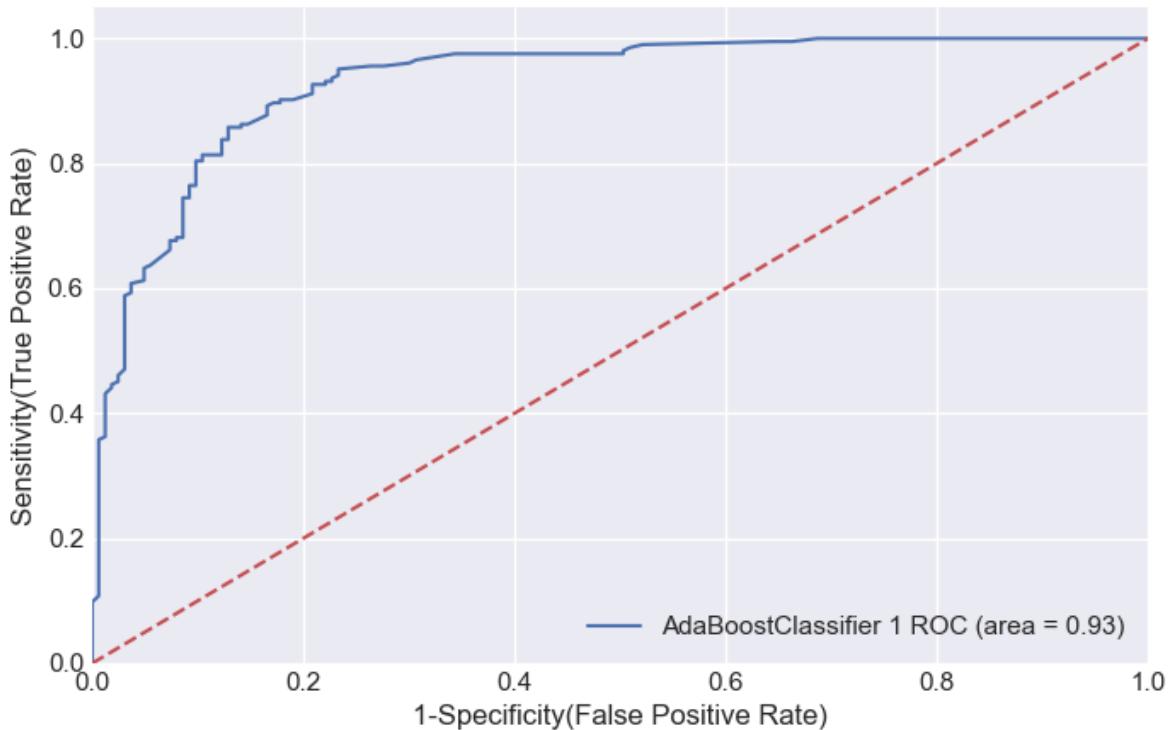
© Dr. Alexander Wagner. Все права охраняются законом



## График №73. Confusion Matrix

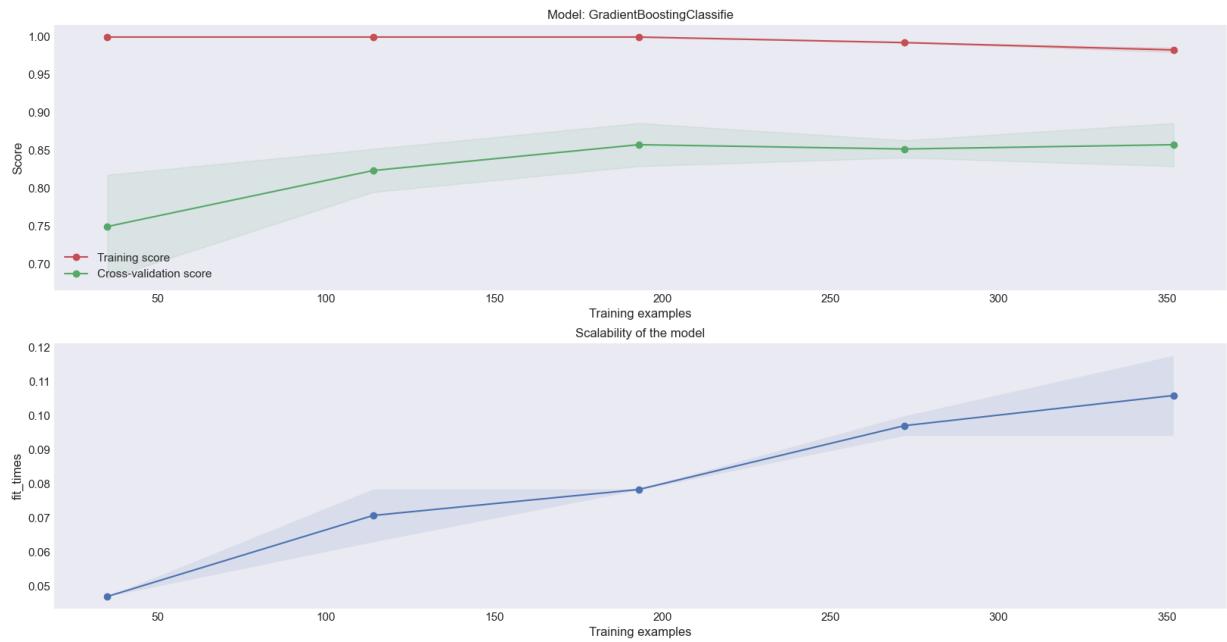


## График №74. ROC Curve





## График №75. Score plot



## Модель: GradientBoostingClassifier

Thanks to <https://www.kaggle.com/kabure/titanic-eda-model-pipeline-keras-nn>

Gradient Boosting builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage `n_classes_` regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced. The features are always randomly permuted at each split. Therefore, the best found split may vary, even with the same training data and `max_features=n_features`, if the improvement of the criterion is identical for several splits enumerated during the search of the best split. To obtain a deterministic behaviour during fitting, `random_state` has to be fixed. Reference sklearn documentation.

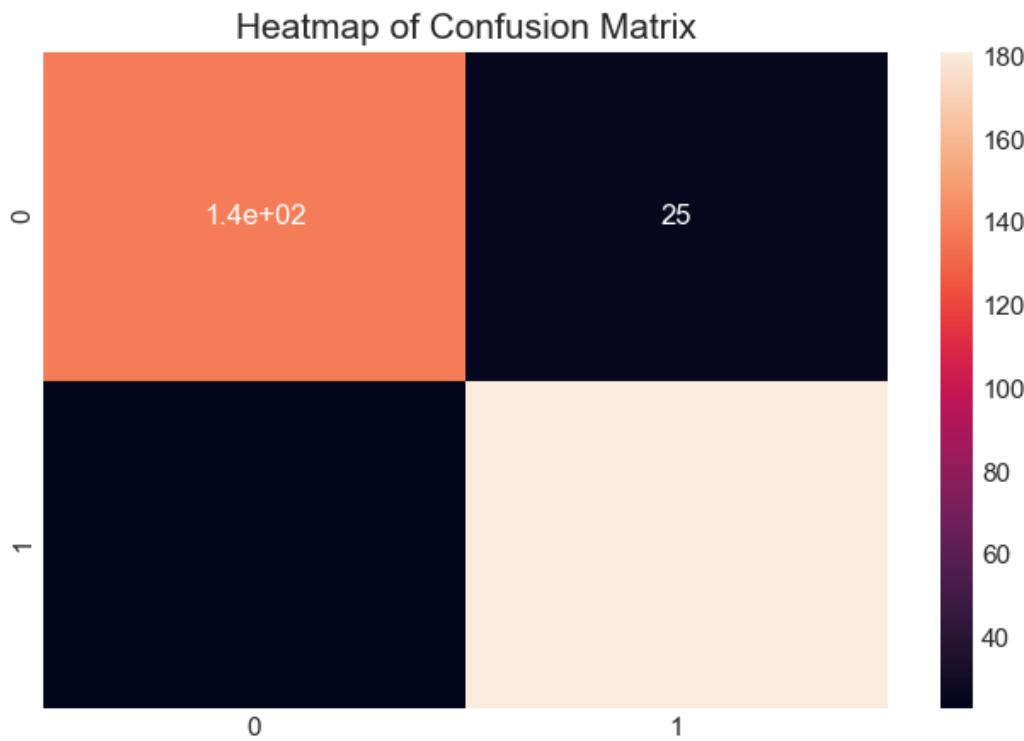
## Таблица №25. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.856	0.835	0.846	164.0
class 1	0.87	0.887	0.878	203.0
accuracy	0.864	0.864	0.864	0.864
macro avg	0.863	0.861	0.862	367.0
weighted avg	0.864	0.864	0.864	367.0

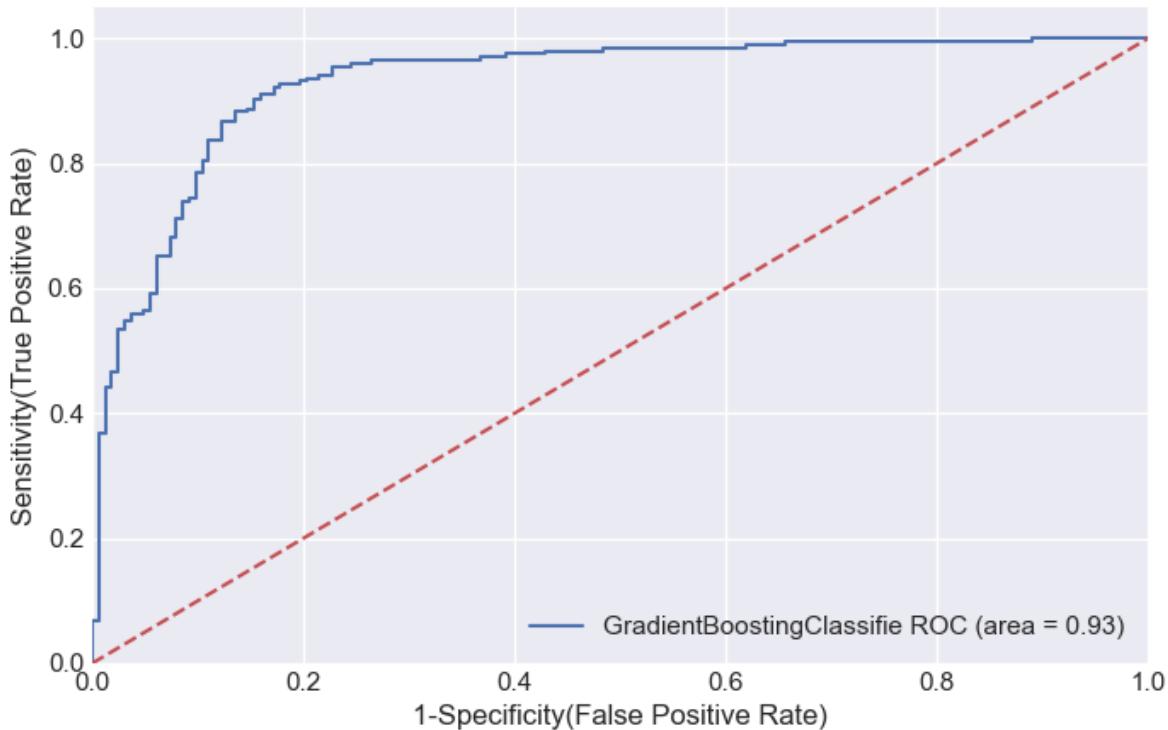
© Dr. Alexander Wagner. Все права охраняются законом



## График №76. Confusion Matrix

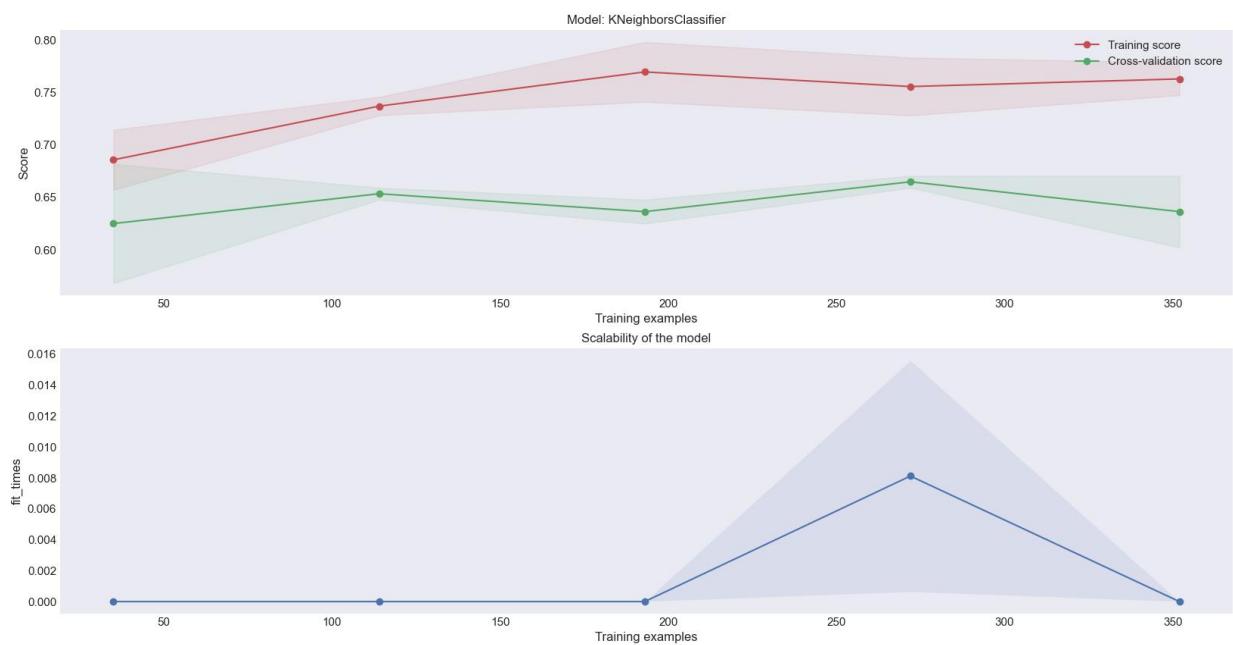


## График №77. ROC Curve





## График №78. Score plot



## Модель: KNeighborsClassifier

Thanks to <https://www.kaggle.com/startupsci/titanic-data-science-solutions>

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. A sample is classified by a majority vote of its neighbors, with the sample being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). Reference Wikipedia.

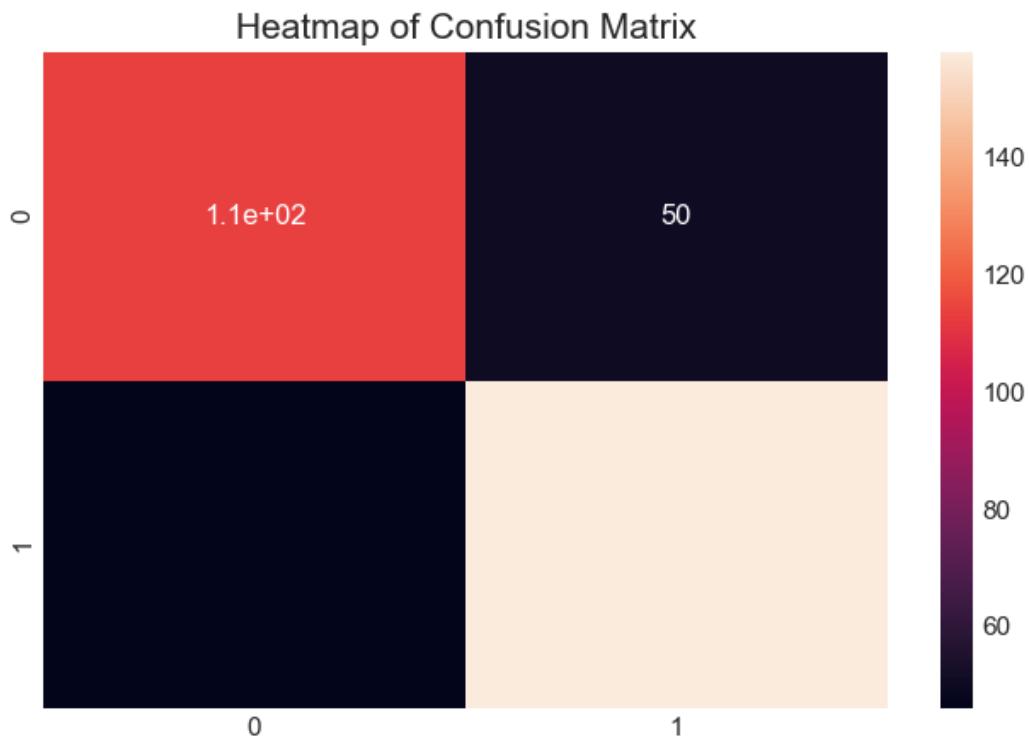
## Таблица №26. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.662	0.61	0.635	164.0
class 1	0.704	0.749	0.726	203.0
accuracy	0.687	0.687	0.687	0.687
macro avg	0.683	0.679	0.68	367.0
weighted avg	0.685	0.687	0.685	367.0

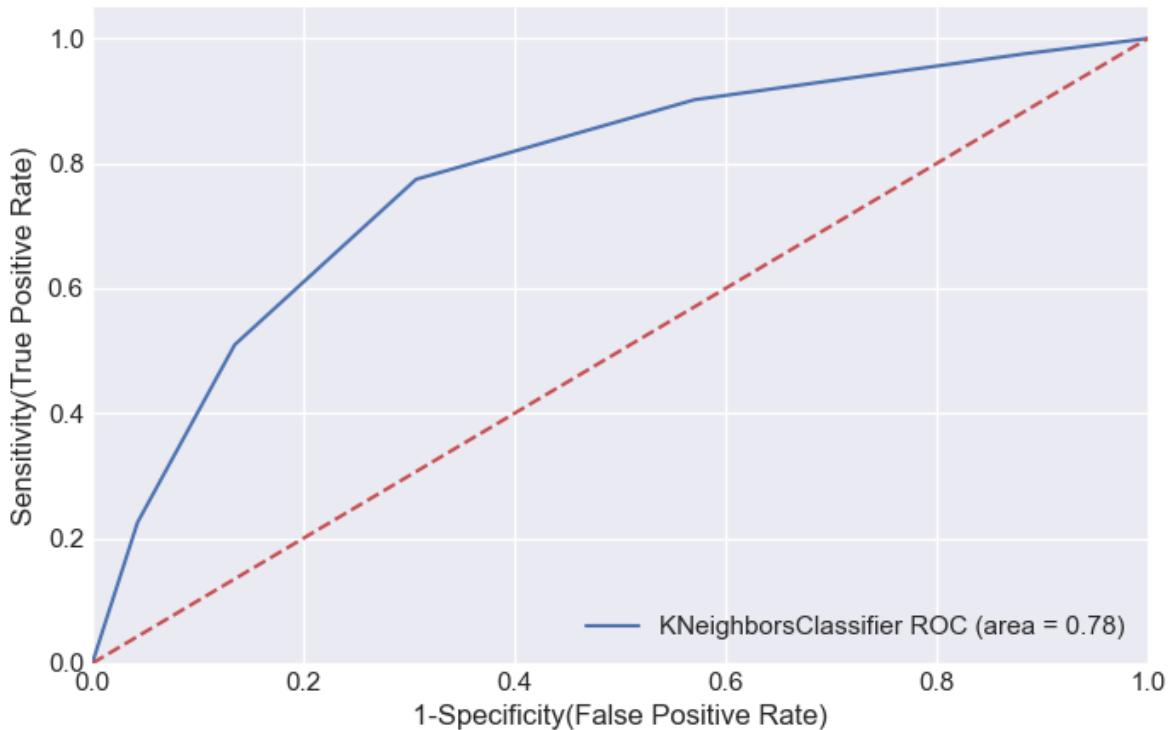
© Dr. Alexander Wagner. Все права охраняются законом



## График №79. Confusion Matrix

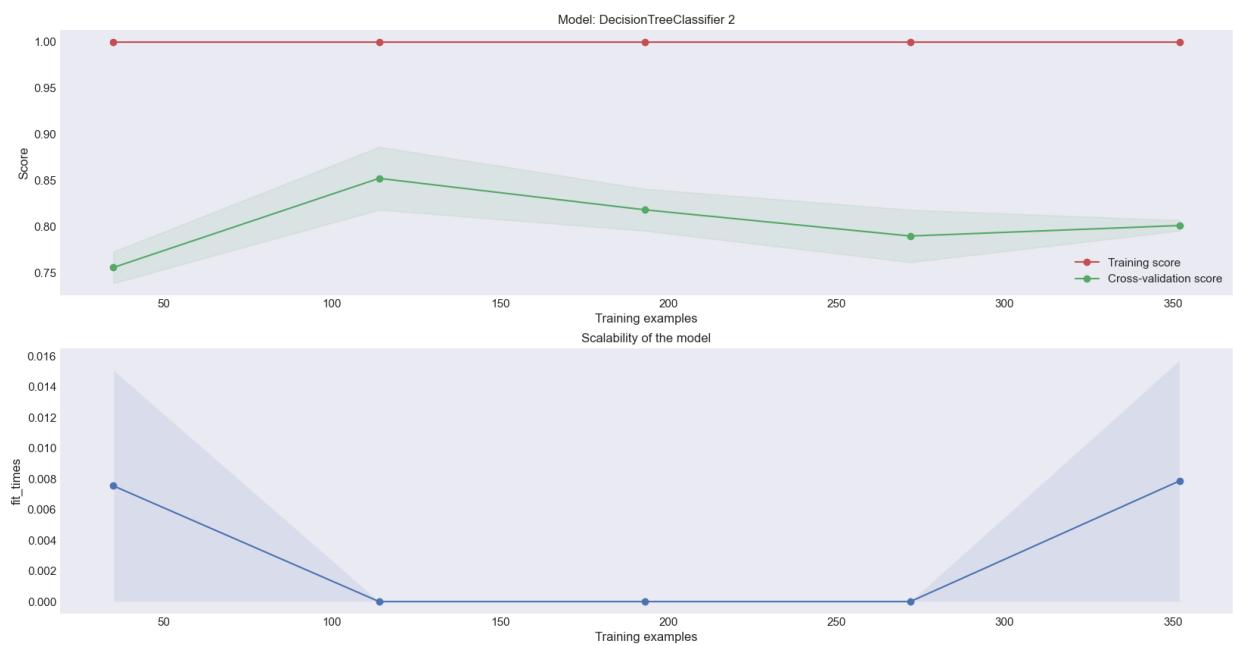


## График №80. ROC Curve





## График №81. Score plot



## Модель: DecisionTreeClassifier 2

This model uses a Decision Tree as a predictive model which maps features (tree branches) to conclusions about the target value (tree leaves). Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Reference Wikipedia.

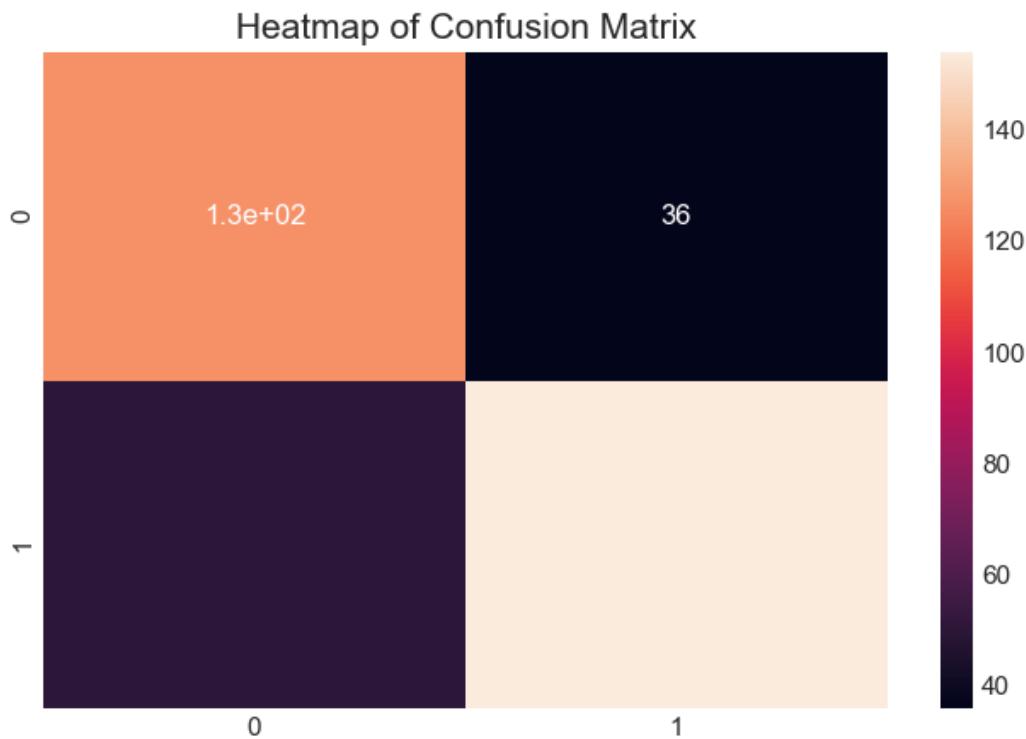
Таблица №27. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.776	0.78	0.778	164.0
class 1	0.822	0.818	0.82	203.0
accuracy	0.801	0.801	0.801	0.801
macro avg	0.799	0.799	0.799	367.0
weighted avg	0.801	0.801	0.801	367.0

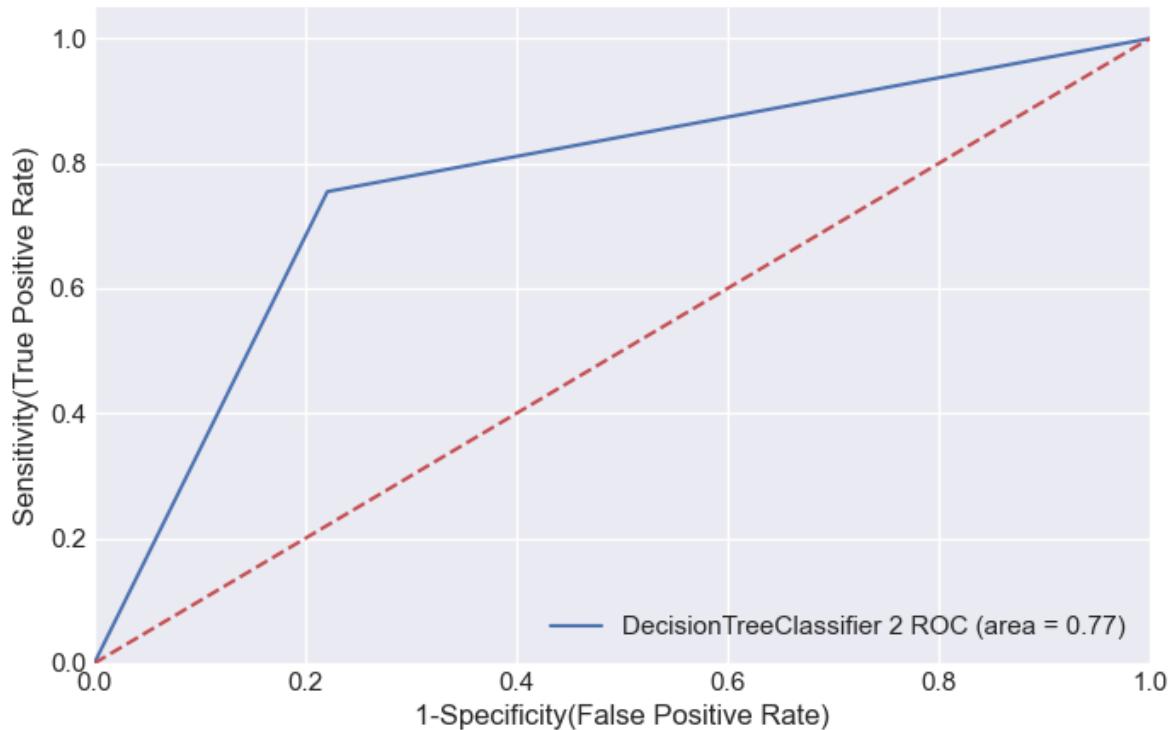
© Dr. Alexander Wagner. Все права охраняются законом



## График №82. Confusion Matrix

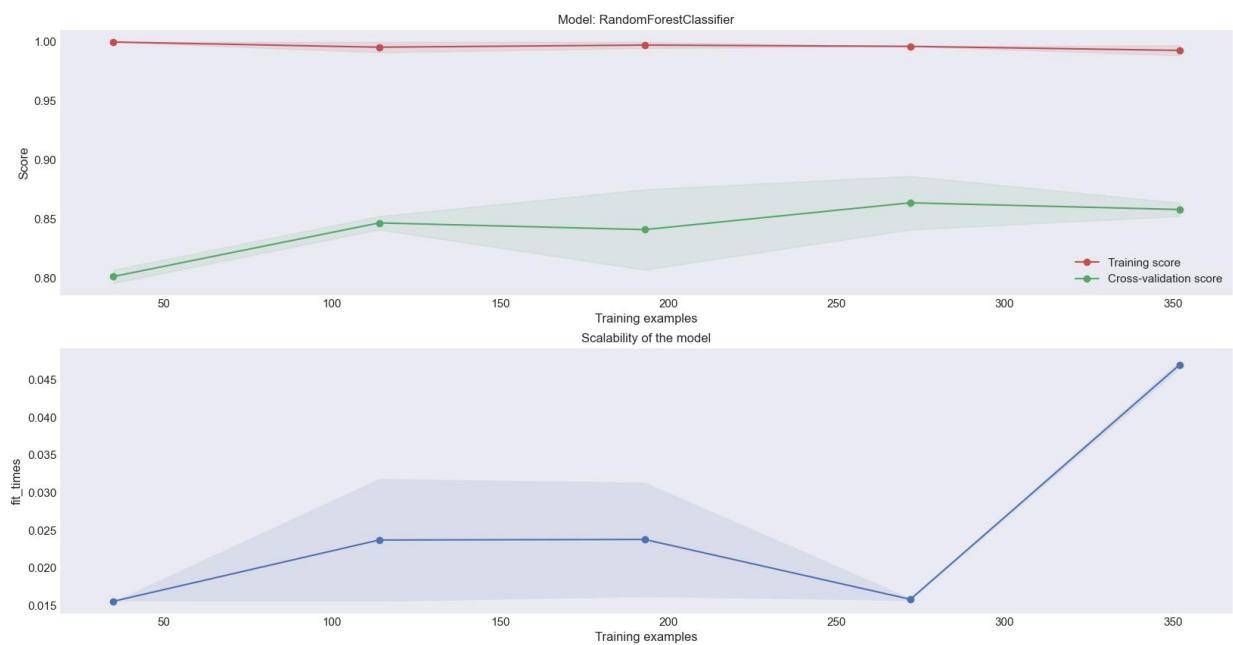


## График №83. ROC Curve





## График №84. Score plot



## Модель: RandomForestClassifier

Random Forest is one of the most popular model. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees (n\_estimators= [100, 300]) at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Reference Wikipedia.

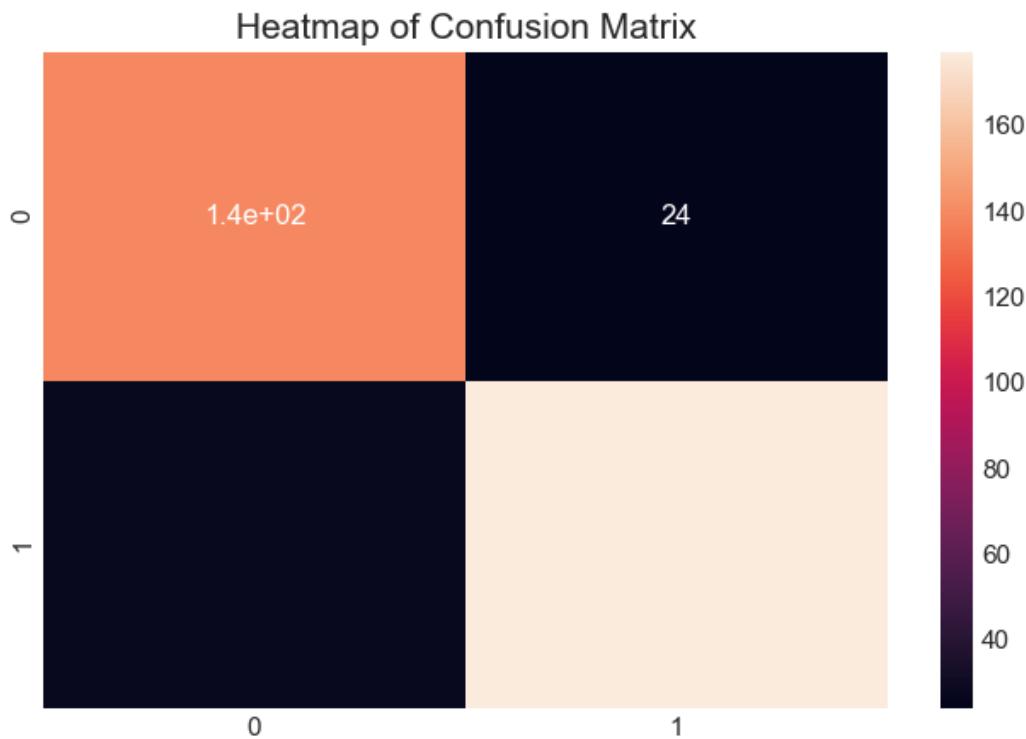
Таблица №28. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.825	0.835	0.83	164.0
class 1	0.866	0.857	0.861	203.0
accuracy	0.847	0.847	0.847	0.847
macro avg	0.845	0.846	0.846	367.0
weighted avg	0.848	0.847	0.847	367.0

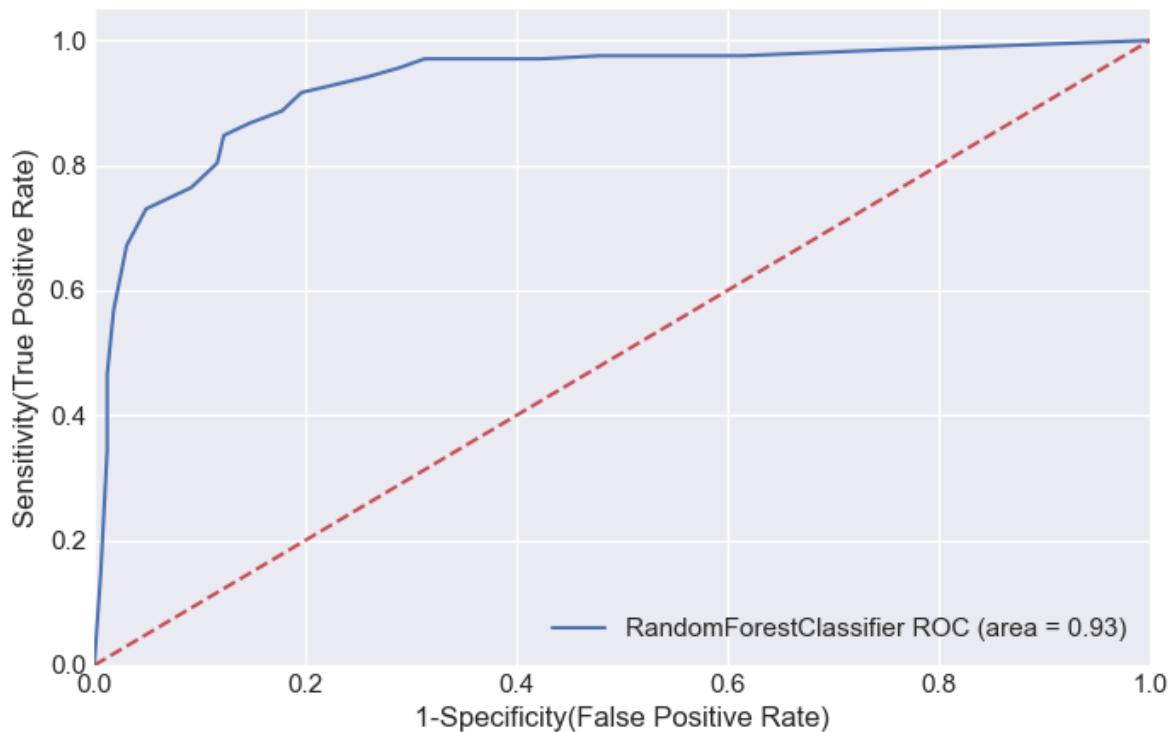
© Dr. Alexander Wagner. Все права охраняются законом



## График №85. Confusion Matrix

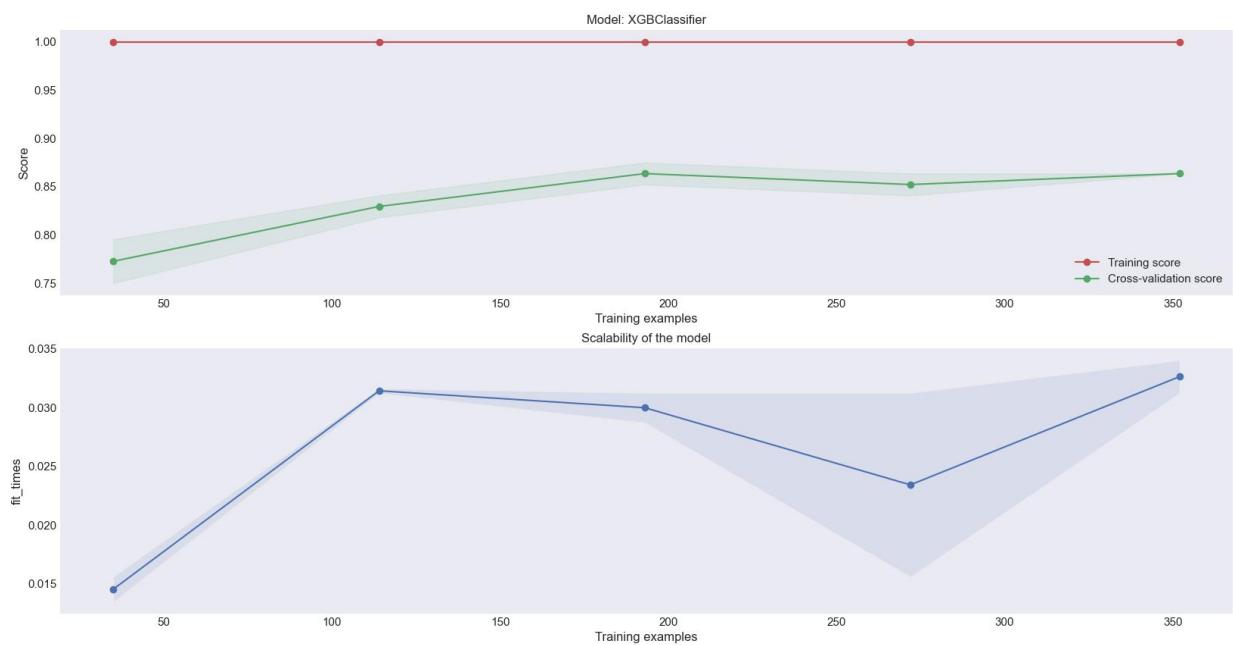


## График №86. ROC Curve





## График №87. Score plot



## Модель: XGBClassifier

XGBoost is an ensemble tree method that applies the principle of boosting weak learners (CARTs generally) using the gradient descent architecture. XGBoost improves upon the base Gradient Boosting Machines (GBM) framework through systems optimization and algorithmic enhancements. Reference Towards Data Science.

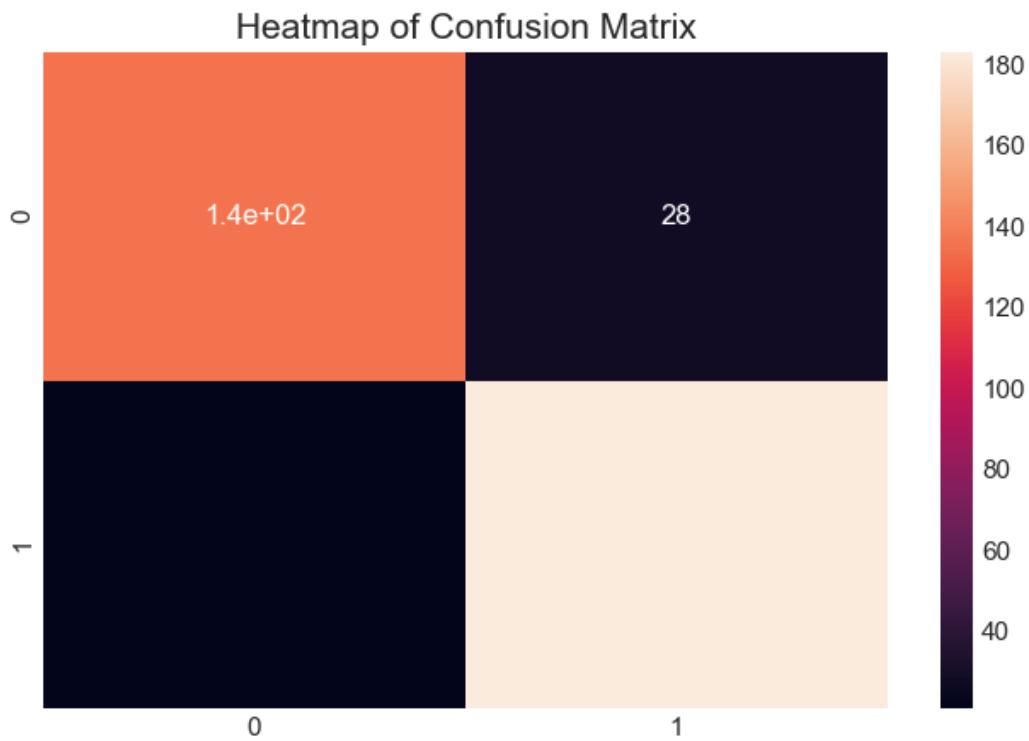
## Таблица №29. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.834	0.829	0.832	164.0
class 1	0.863	0.867	0.865	203.0
accuracy	0.85	0.85	0.85	0.85
macro avg	0.849	0.848	0.848	367.0
weighted avg	0.85	0.85	0.85	367.0

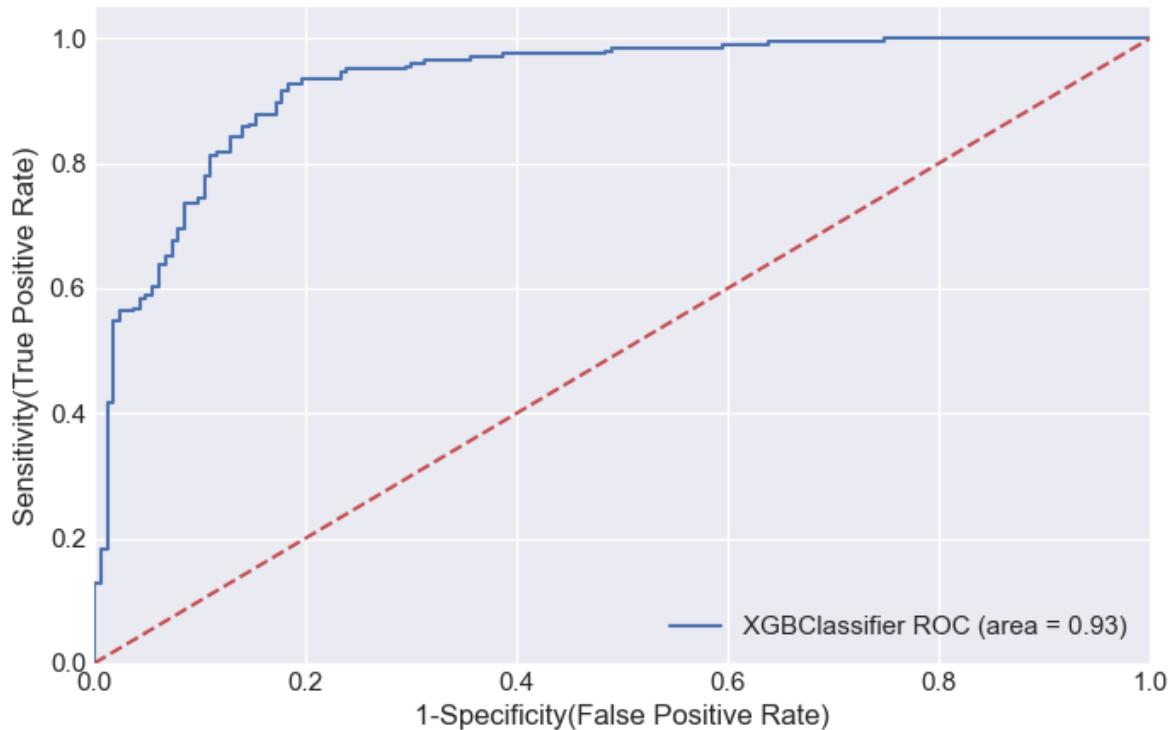
© Dr. Alexander Wagner. Все права охраняются законом



## График №88. Confusion Matrix

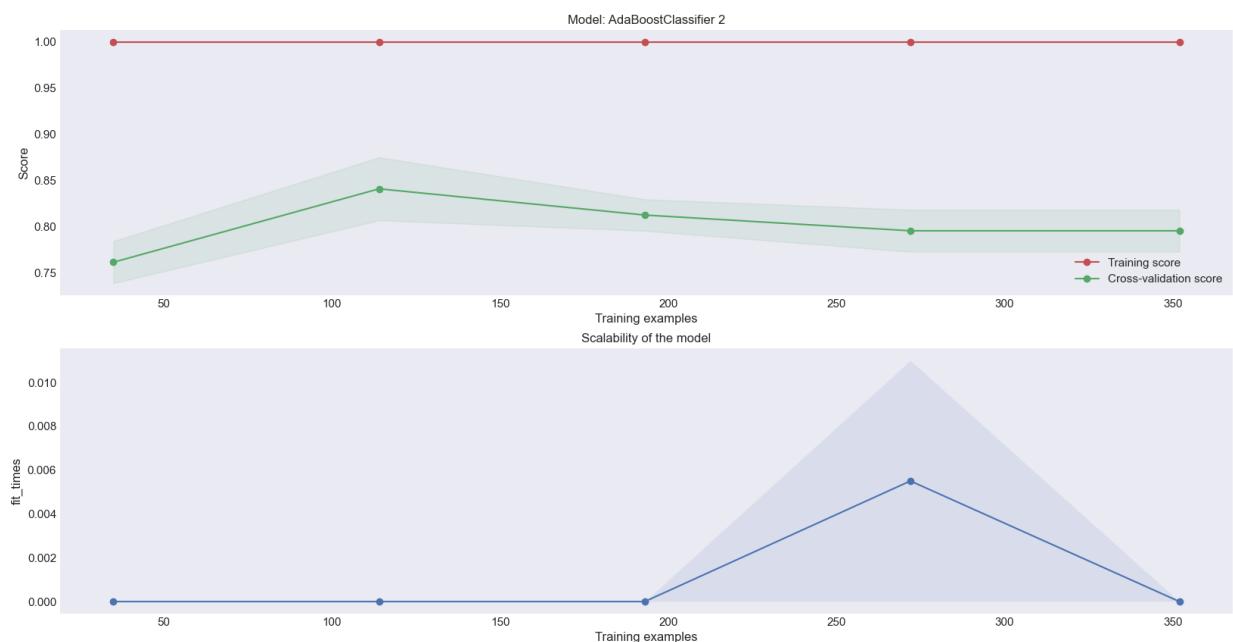


## График №89. ROC Curve





## График №90. Score plot



## Модель: AdaBoostClassifier 2

The core principle of AdaBoost ("Adaptive Boosting") is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying N weights to each of the training samples. Initially, those weights are all set to  $1/N$ , so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence. Reference sklearn documentation.

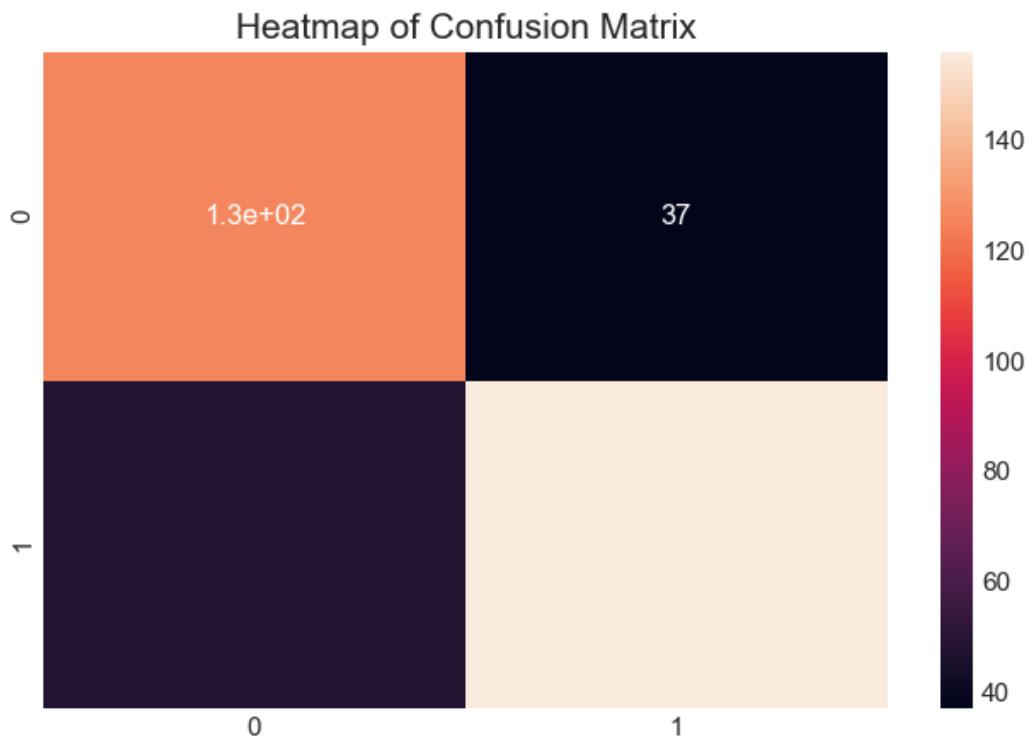
## Таблица №30. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.794	0.823	0.808	164.0
class 1	0.853	0.828	0.84	203.0
accuracy	0.826	0.826	0.826	0.826
macro avg	0.823	0.825	0.824	367.0
weighted avg	0.827	0.826	0.826	367.0

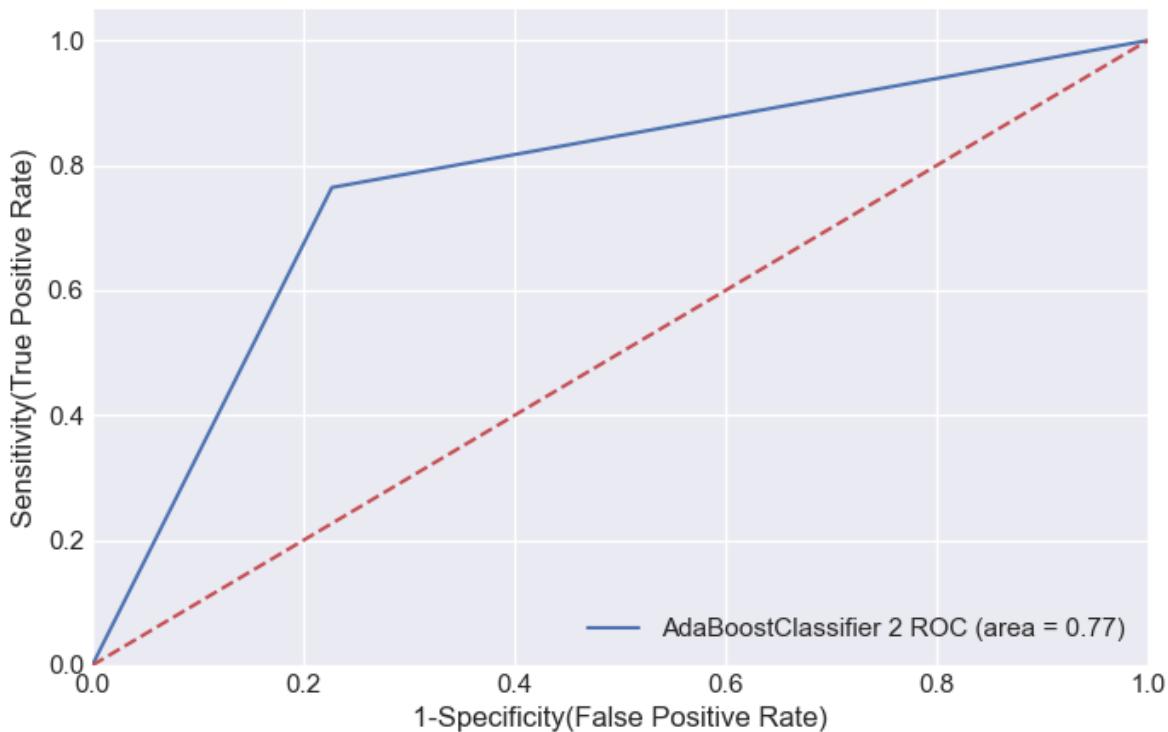
© Dr. Alexander Wagner. Все права охраняются законом



## График №91. Confusion Matrix

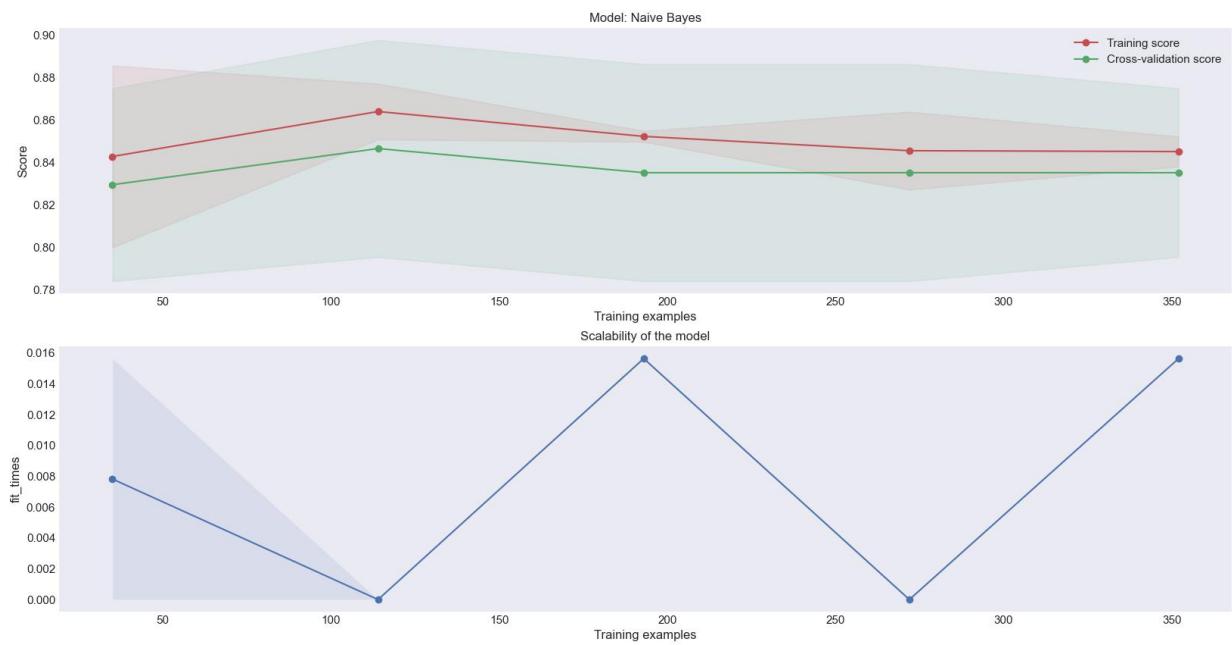


## График №92. ROC Curve





## График №93. Score plot



## Модель: Naive Bayes

Thanks to <https://www.kaggle.com/startupsci/titanic-data-science-solutions>

In machine learning, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features) in a learning problem. Reference Wikipedia.

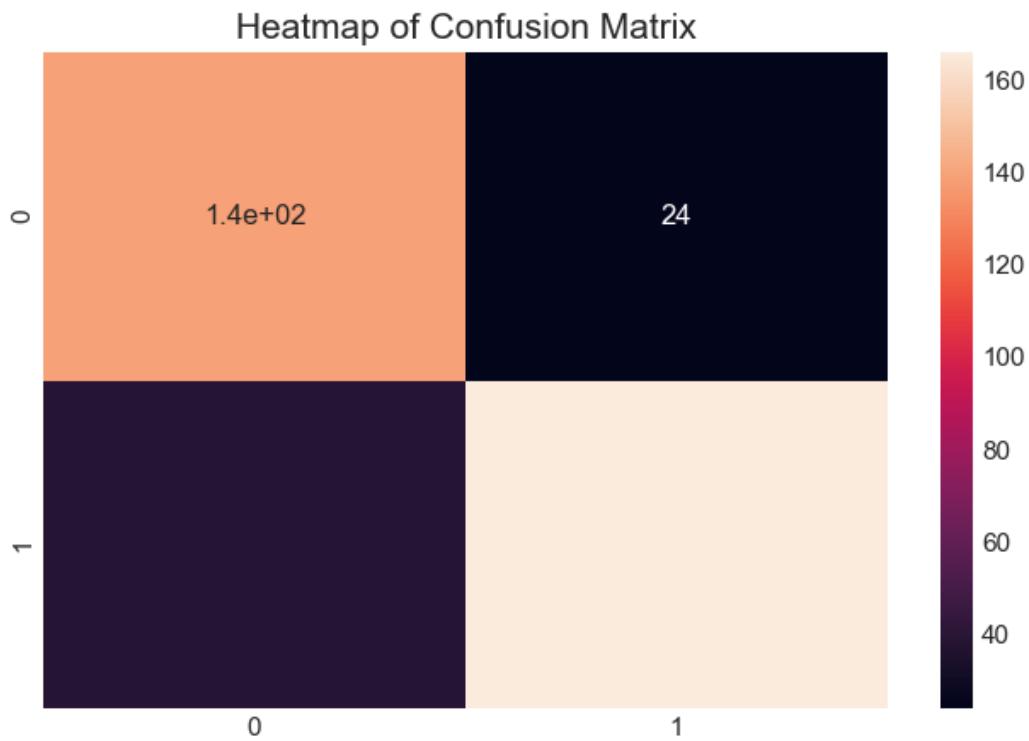
## Таблица №31. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.811	0.835	0.823	164.0
class 1	0.864	0.842	0.853	203.0
accuracy	0.839	0.839	0.839	0.839
macro avg	0.837	0.839	0.838	367.0
weighted avg	0.84	0.839	0.839	367.0

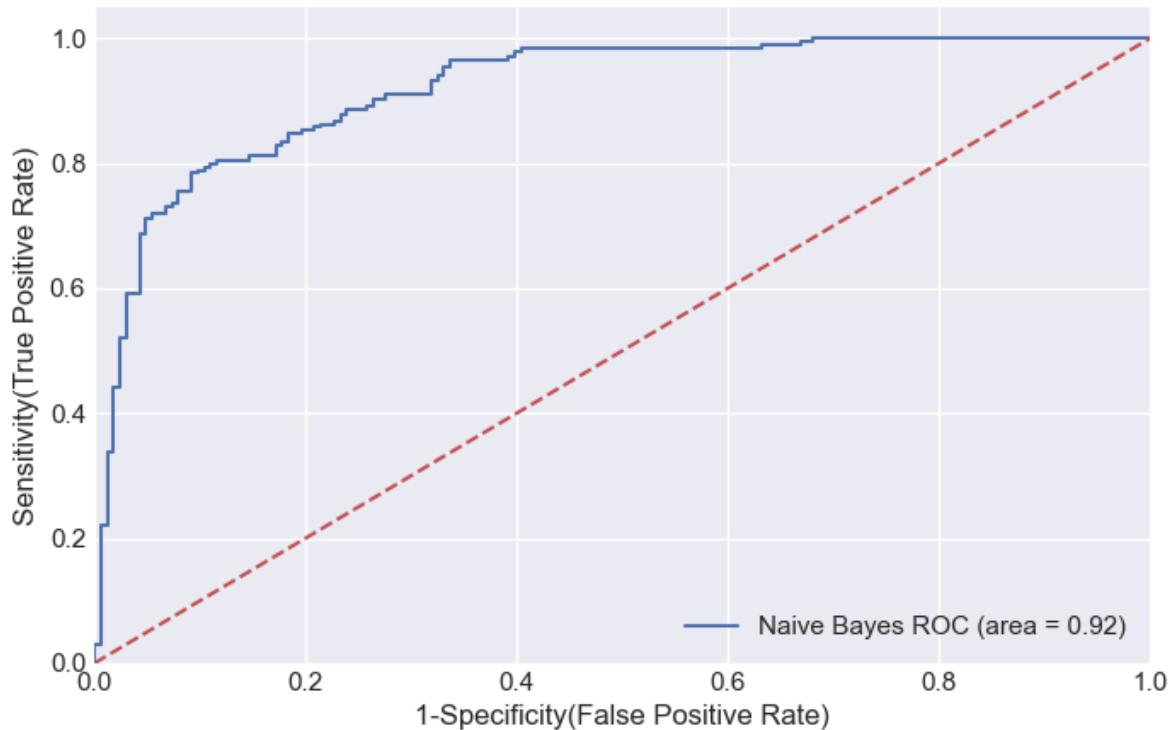
© Dr. Alexander Wagner. Все права охраняются законом



## График №94. Confusion Matrix

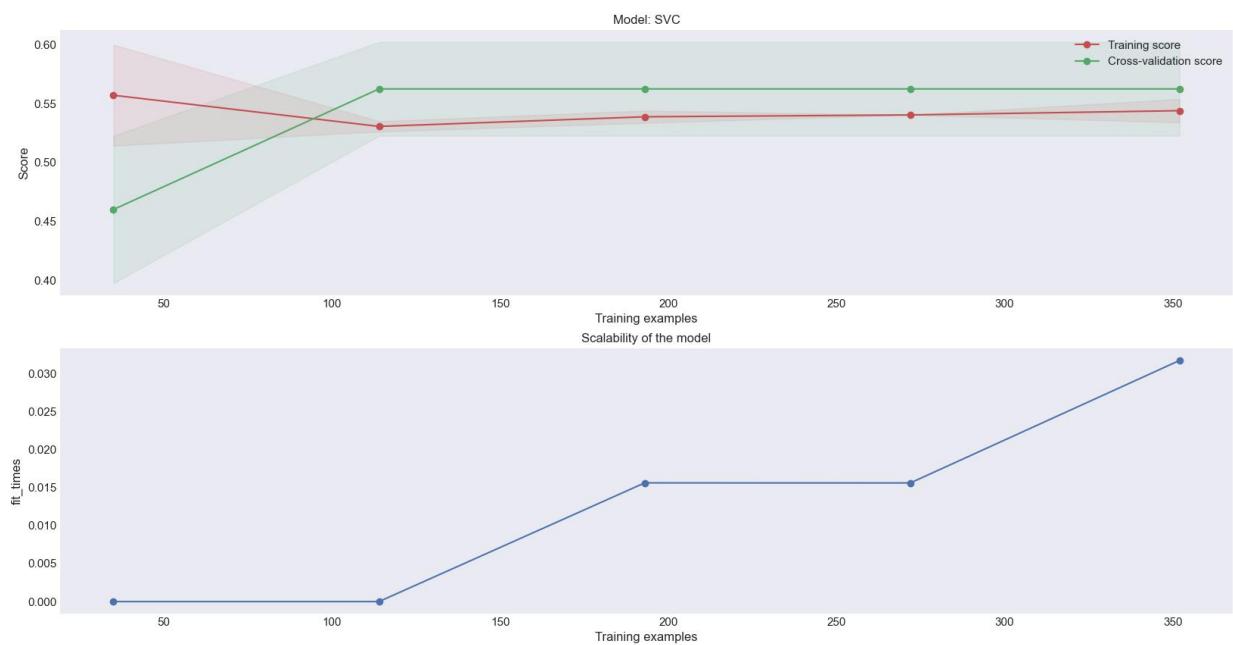


## График №95. ROC Curve





## График №96. Score plot



## Модель: SVC

Support Vector Machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training samples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new test samples to one category or the other, making it a non-probabilistic binary linear classifier. Reference Wikipedia.

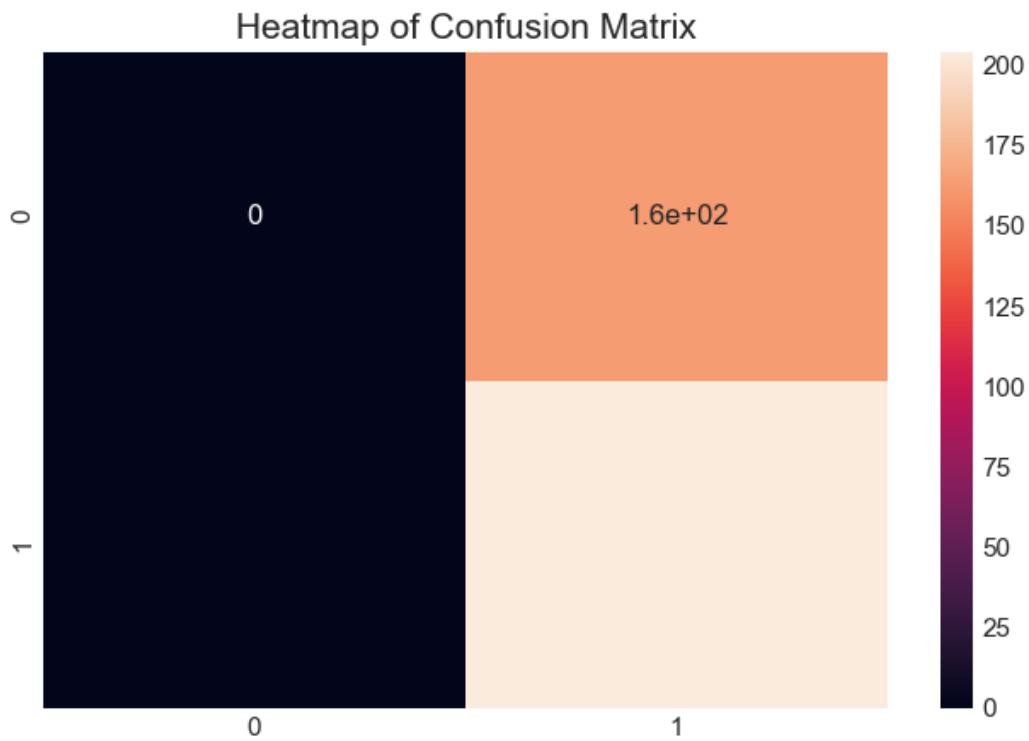
Таблица №32. Таблица классификации

Classes+Metrics	precision	recall	f1-score	support
class 0	0.0	0.0	0.0	164.0
class 1	0.553	1.0	0.712	203.0
accuracy	0.553	0.553	0.553	0.553
macro avg	0.277	0.5	0.356	367.0
weighted avg	0.306	0.553	0.394	367.0

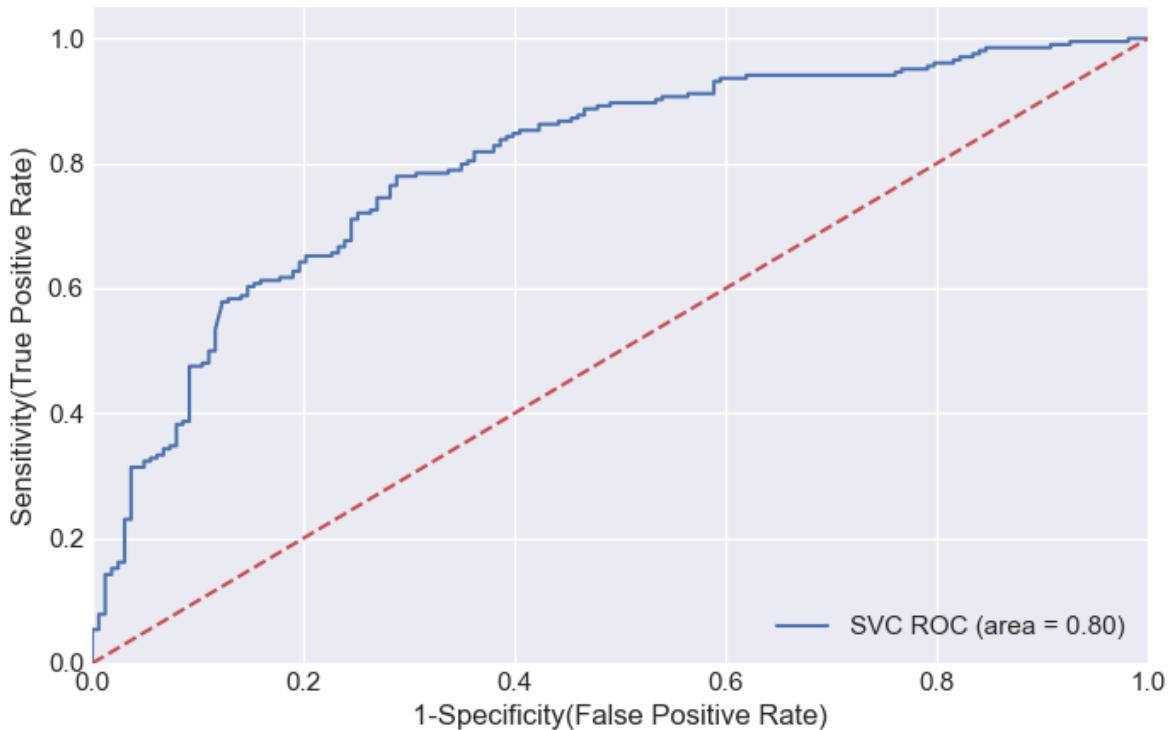
© Dr. Alexander Wagner. Все права охраняются законом



## График №97. Confusion Matrix

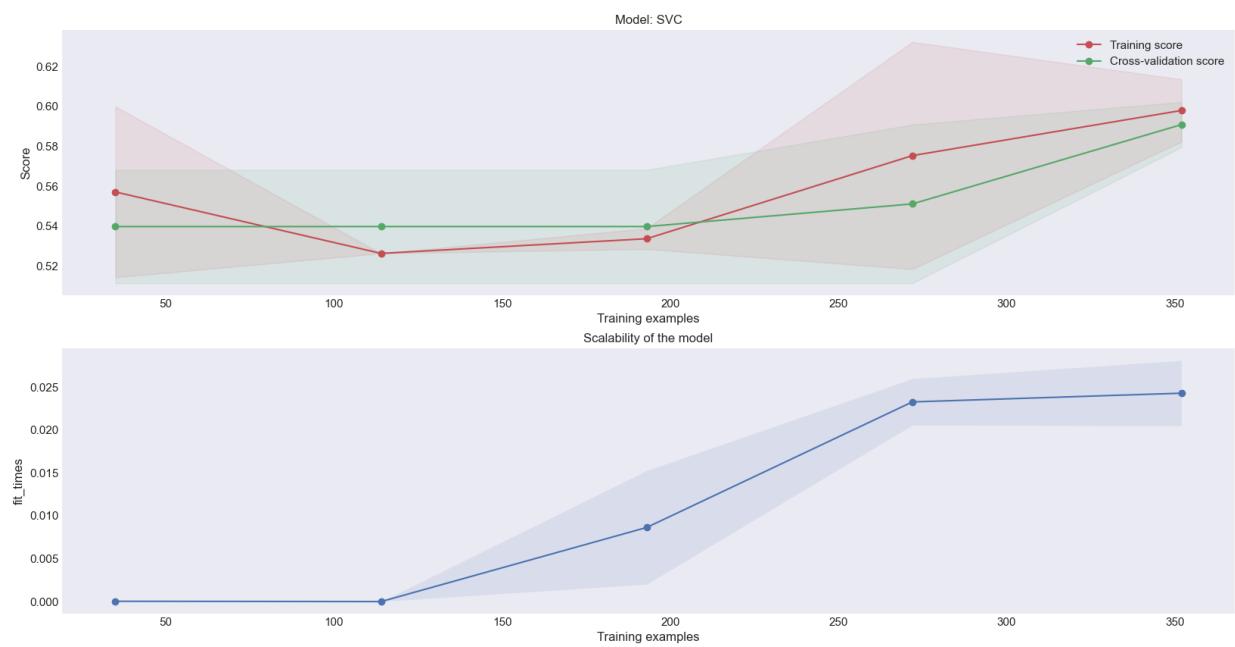


## График №98. ROC Curve





## График №99. Score plot





## Результаты моделирования

Моделирование осуществлялось при помощи методов машинного обучения. Для исследования проблемы были выбраны 18 моделей в том числе:

- Linear Regression
- Logistic Regression
- Perceptron
- Linear SVC
- MLPClassifier
- Decision Tree Classifier 1
- Stochastic Gradient Decent
- RidgeClassifier
- BaggingClassifier
- AdaBoostClassifier 1
- GradientBoostingClassifier
- KNeighborsClassifier
- DecisionTreeClassifier 2
- RandomForestClassifier
- XGBClassifier
- AdaBoostClassifier 2
- Naive Bayes
- SVC

В результате работы каждой модели получена следующая информация в форме таблиц и графиков:

- Таблица классификация
- Матрица Confusion Matrix
- ROC график
- Score график

Таблица классификация даёт оценку модели по критериям:

- *precision*
- *recall*
- *f1-score*

ROC график даёт оценку модели по критерию AUC, т.е. значению площади под кривой. Что означает – чем ближе этот показатель к значению равному 1, тем выше соответствие прогнозных значений модели реальным входным данным.



## Оценка моделей и выбор наилучших для использования в диагностике ССЗ

В результате анализа и моделирования получена оценка исходных данных и характеристики для всех моделей. Итоговые результаты представлены в виде таблиц и графиков, объединенный график ROC приведен рис. №100, обобщенные таблицы №33-№35 показывают сравнительные характеристики моделей.

График №100. ROC-график для всех моделей

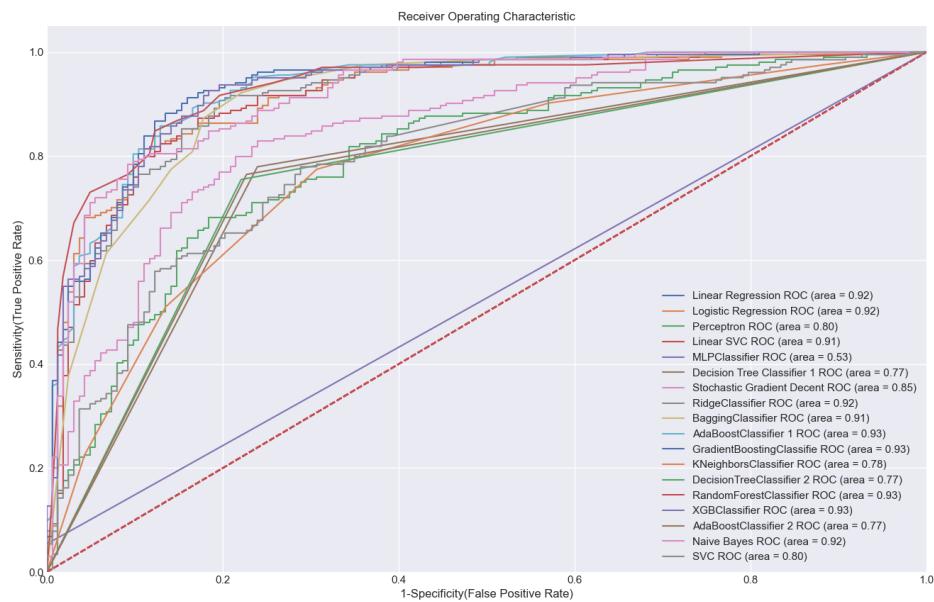


График №101 r2\_score %. Линейный график А для всех моделей

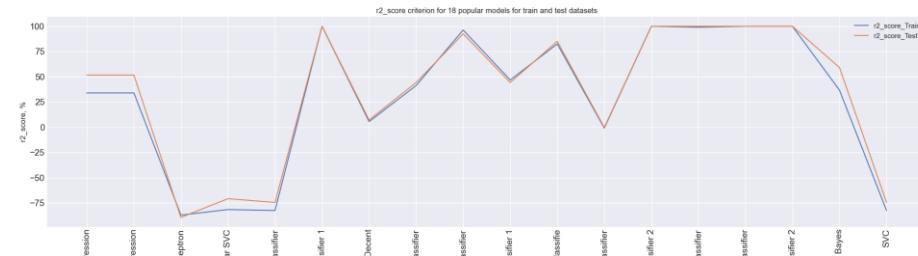
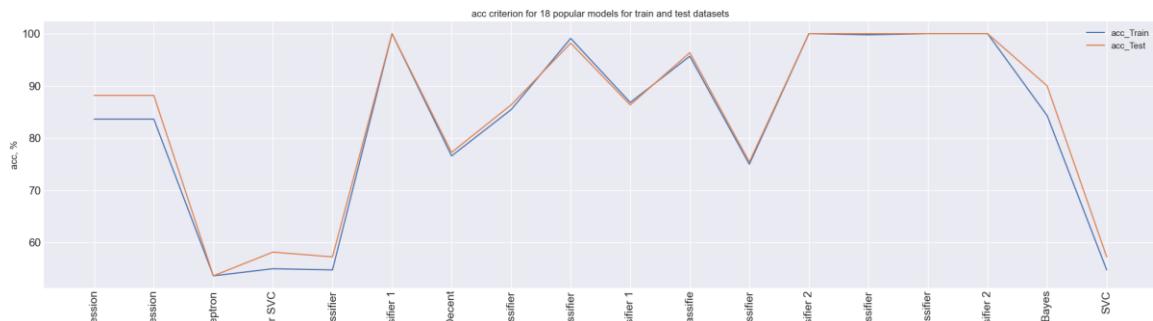


График №102 ACC%. Линейный график В для всех моделей





## График №103 rmse %. Линейный график С для всех моделей

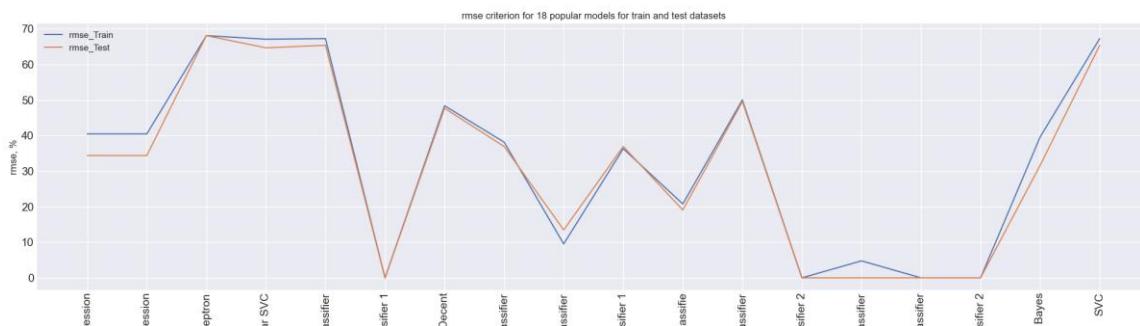


Table №33. Характеристика лучших моделей после первого этапа

Model	r2_sco	r2_score_	acc_Tra	acc_Tes	acc_Diff	rmse_Tr	rmse_	re_Trai	re_Test
Decision Tree	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0
DecisionTreeCla	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0
XGBClassifier	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0
AdaBoostClass2	100.0	100.0	100.0	100.0	0.0	0.0	0.0	0.0	0.0
RandomForestCl	99.08	100.0	99.77	100.0	-	4.77	0.0	0.42	0.0
BaggingClassifie	95.42	88.64	98.86	97.27	1.59000	10.66	16.51	2.09	4.55
GradientBoostin	81.68	88.64	95.45	97.27	-	21.32	16.51	8.37	4.55
AdaBoostClass1	62.45	69.7	90.68	92.73	-	30.53	26.97	17.15	12.12

© Dr. Alexander Wagner. Все права охраняются законом

Table №34. Характеристика всех моделей после второго этапа

Model	acc_train
RandomForestClas	100.0
Decision Tree	100.0
DecisionTreeClass	100.0
BaggingClassifier	99.09
GradientBoosting	95.82
XGBClassifier	91.27
AdaBoostClassifie	91.09
RidgeClassifier	85.09
Logistic	84.73
Linear Regression	84.18
KNeighborsClassif	78.18
Linear SVC	55.45
SVC	55.45
Perceptron	54.36
Stochastic	49.27
MLPClassifier	44.55

© Dr. Alexander Wagner. Все права охраняются законом

Table №35. Характеристика всех моделей после третьего этапа

Model	r2_score_train	acc_train	rmse_train	re_train
RandomForestClas	100.0	100.0	0.0	0.0
Decision Tree	100.0	100.0	0.0	0.0
DecisionTreeClass	100.0	100.0	0.0	0.0

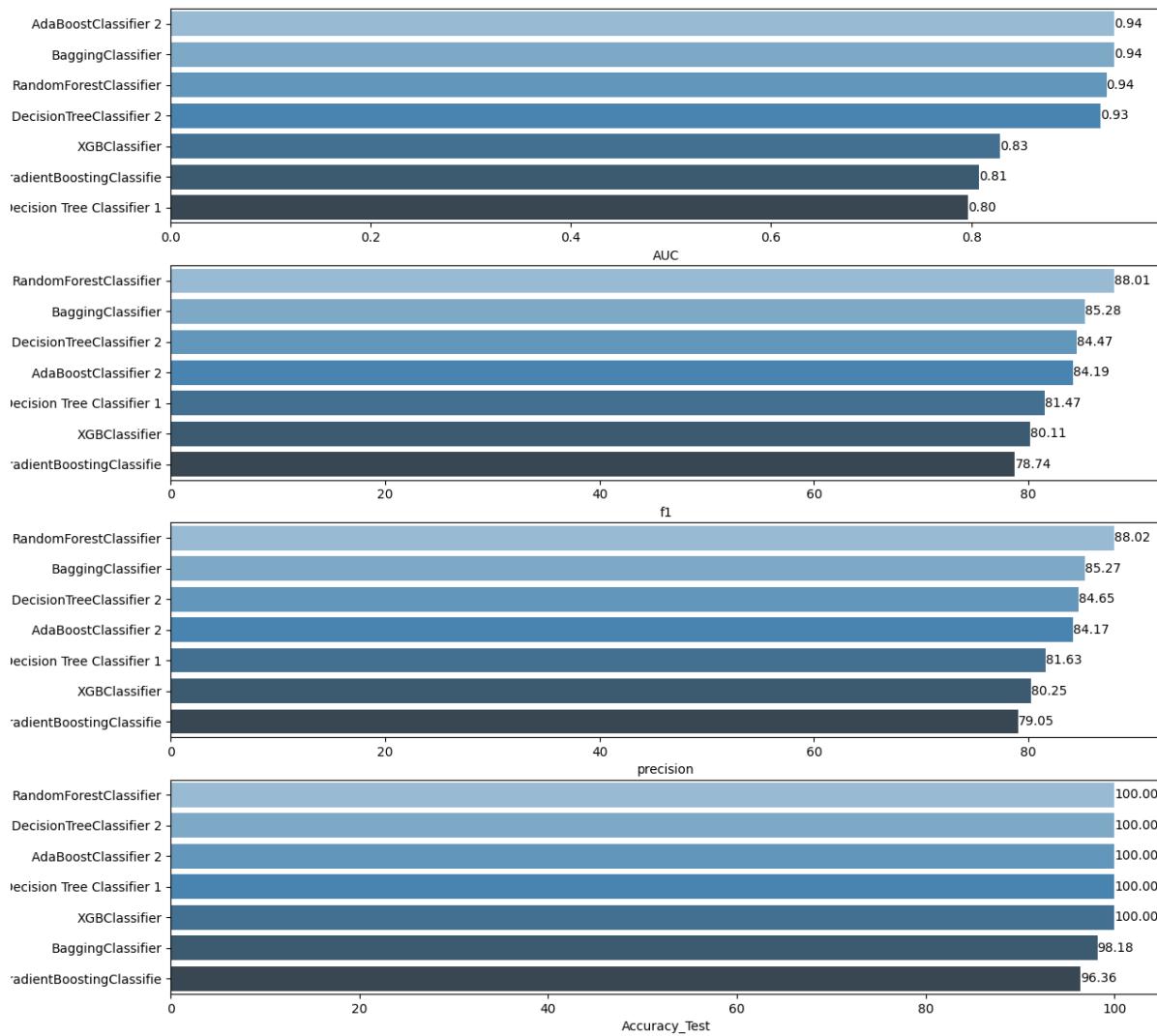


Model	r2_score_train	acc_train	rmse_train	re_train
BaggingClassifier	96.32	99.09	9.53	1.64
GradientBoosting	83.07	95.82	20.45	7.54
XGBClassifier	64.67	91.27	29.54	15.74
AdaBoostClassifie	63.93	91.09	29.85	16.07
RidgeClassifier	39.65	85.09	38.61	26.89
Logistic	38.17	84.73	39.08	27.54
Linear Regression	35.97	84.18	39.77	28.52
KNeighborsClassif	11.68	78.18	46.71	39.34
Linear SVC	-80.33	55.45	66.74	80.33
SVC	-80.33	55.45	66.74	80.33
Perceptron	-84.74	54.36	67.55	82.3
Stochastic	-105.35	49.27	71.22	91.48
MLPClassifier	-124.49	44.55	74.47	100.0

© Dr. Alexander Wagner. Все права охраняются законом

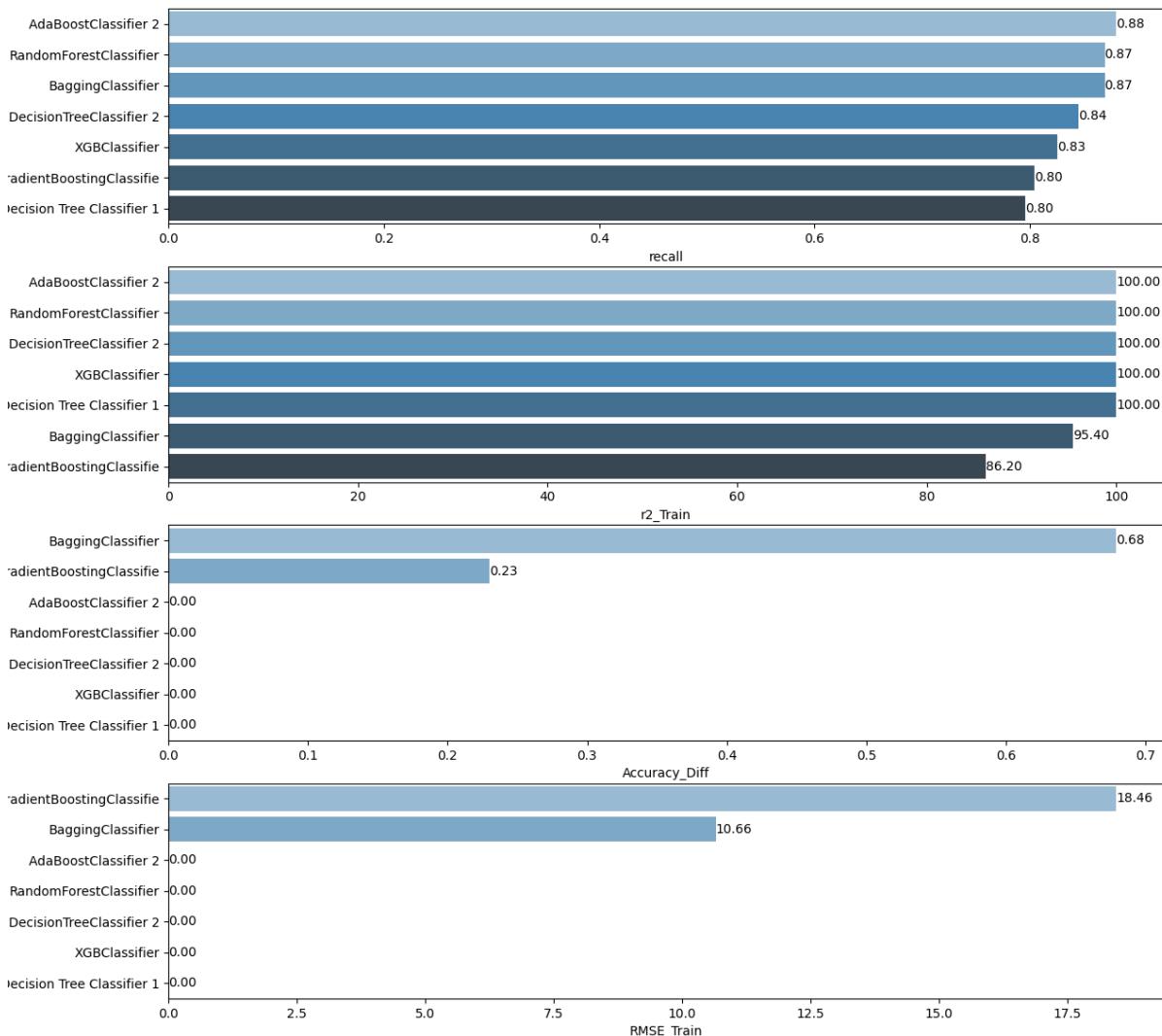


## График №104. График основных метрик для лучших моделей: AUC, F1, Precision, Accuracy\_Test





## График №105. График основных метрик для лучших моделей: Recall, r2\_Train, Accuracy\_Diff, RMSE\_Train





## Заключение

Системы и модели поддержки принятия решений на основе машинного обучения для раннего прогнозирования и диагностики находят широкое применение в здравоохранении. Эти системы помогают пациентам и медицинскому персоналу улучшить процесс принятия решений и раннее прогнозирование возникновения МАСЕ у пациентов с острым инфарктом миокарда. По сравнению с другими известными алгоритмами и системами прогнозирования, мы обнаружили, что алгоритмы машинного обучения лучше работают в прогнозировании и диагностике МАСЕ.

Лучшими алгоритмами машинного обучения в нашем исследовании стали:

1. Decision Tree Classifier 1
2. DecisionTreeClassifier 2
3. RandomForestClassifier
4. XGBClassifier
5. AdaBoostClassifier 2
6. BaggingClassifier
7. GradientBoostingClassifier

Другие модели прогнозирования, основанные на машинном обучении, также были протестированы, но они показали худшие результаты, а их точность была меньше, чем у этих моделей, поэтому эти 7 моделей были доработаны в нашем исследования и рекомендуются к применению к использованию или проверки в аналогичных исследованиях.

В данной статье мы применили алгоритмы машинного обучения для раннего прогнозирования ССЗ у кардио-пациентов. Производительность этих моделей была сравнена с нашей моделью ансамбля мягкого голосования на основе машинного обучения. По результатам эксперимента мы обнаружили, что производительность нашего классификатора ансамбля мягкого голосования превзошла производительность других моделей машинного обучения. Кроме того, прогностические факторы для классификатора ансамбля мягкого голосования отличались от регрессионных моделей.



## **Рекомендации по применению АСНИ для исследователей**

Исследователям, желающим осуществить свое научное исследование самостоятельно с использованием настоящей АСНИ рекомендуется ознакомиться с её применением на специальном курсе, который автор готов провести онлайн или офлайн при наличие интереса и по согласованию с руководством Кафедры. **Курс бесплатный.** Возможны индивидуальные или групповые консультации по согласованию.

Для использования АСНИ пользователям не требуется никакие дополнительные программные средства. Стандартная версия работает в Windows(e) и использует лишь некобходимые продукты (Word, Excel) Microsoft Office последней версии.



## **Актуальное состояние и дальнейшее развитие АСНИ**

В настоящий момент АСНИ прошла опытную эксплуатацию и работает в локальном режиме. Платформа для размещения системы Web-Servere определена, программные средства и режим работы частично протестированы и показали их стабильное функционирование (пока в тестовом варианте).

Автор планирует после полного завершения разработки АСНИ разместить её на Web-Servere для свободного доступа всем желающим. Ориентировочная дата готовности серверного варианта: IV квартал 2024 года.

В новой версии Системы будут устранены имеющиеся ограхи, неточности и ошибки. Будут добавлены новые функциональные блоки и оптимизированы существующие.

Научное исследование проведено на базе открытых данных с использованием типовых методов Анализа, описание этого исследования проведено в форме доступной для понимания специалиста нематематика (неинформатика), обладающего опытом работы с современными Информационными системами общего назначения.

### **Примечание:**

- В разработку данной системы автор инвестировал по минимальной оценке 1200 часов интенсивной работы
- Для дальнейшей разработки этой системы потребуется 6-8 календарных месяцев и ориентировочно 1000 рабочих часов
- В приложении для профессиональных разработчиков и любознательных пользователей в качестве иллюстрации и представления приведены фрагменты приограммного кода АСНИ



## Литература

- [1] W. Guan et al., “Clinical characteristics of coronavirus disease 2019 in China,” *New England Journal of Medicine*, 2020, doi: [10.1056/NEJMoa2002032](https://doi.org/10.1056/NEJMoa2002032).
- [2] WHO Team, “Report of the WHO-China joint mission on coronavirus disease 2019 (COVID-19).” Available: [https://www.who.int/publications-detail/report-of-the-who-china-joint-mission-on-coronavirus-disease-2019-\(covid-19\)](https://www.who.int/publications-detail/report-of-the-who-china-joint-mission-on-coronavirus-disease-2019-(covid-19))
- [3] Center for Systems Science and Engineering (CSSE), “COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University.” 2020. Available: <https://github.com/CSSEGISandData/COVID-19>
- [4] J. Textor, B. van der Zander, M. S. Gilthorpe, M. Liśkiewicz, and G. T. Ellison, “Robust causal inference using directed acyclic graphs: The R package ‘dagitty’,” *International Journal of Epidemiology*, vol. 45, no. 6, pp. 1887–1894, Jan. 2017, doi: [10.1093/ije/dyw341](https://doi.org/10.1093/ije/dyw341).
- [5] S. Schipf, S. Knüppel, J. Hardt, and A. Stang, “Directed Acyclic Graphs (DAGs) – Die Anwendung kausaler Graphen in der Epidemiologie,” *Gesundheitswesen*, vol. 73, no. 12, pp. 888–892, Dec. 2011, doi: [10.1055/s-0031-1291192](https://doi.org/10.1055/s-0031-1291192).
- [6] S. Greenland, J. M. Robins, and J. Pearl, “Confounding and Collapsibility in Causal Inference,” *Statistical Science*, vol. 14, no. 1, pp. 29–46, 1999, Available: <http://www.jstor.org/stable/2676645>
- [7] M. Chinazzi et al., “The effect of travel restrictions on the spread of the 2019 novel coronavirus (COVID-19) outbreak,” *Science*, vol. 368, no. 6489, pp. 395–400, 2020, doi: [10.1126/science.aba9757](https://doi.org/10.1126/science.aba9757).
- [8] M. U. G. Kraemer et al., “The effect of human mobility and control measures on the COVID-19 epidemic in China.” *Science (New York, N.Y.)*, vol. 368, no. 6490, pp. 493–497, May 2020, doi: [10.1126/science.abb4218](https://doi.org/10.1126/science.abb4218).
- [9] K. Linka, M. Peirlinck, F. Sahli Costabal, and E. Kuhl, “Outbreak dynamics of COVID-19 in Europe and the effect of travel restrictions.” *Computer methods in biomechanics and biomedical engineering*, pp. 1–8, May 2020, doi: [10.1080/10255842.2020.1759560](https://doi.org/10.1080/10255842.2020.1759560).
- [10] C. Santana, F. Botta, H. Barbosa, F. Privitera, R. Menezes, and R. Di Clemente, “Analysis of human mobility in the UK during the COVID-19 pandemic,” 2020.
- [11] S. Engle, J. Stromme, and A. Zhou, “Staying at home: Mobility effects of COVID-19,” Available at SSRN, 2020, Available: <http://dx.doi.org/10.2139/ssrn.3565703>
- [12] M. Mazzoli, D. Mateo, A. Hernando, S. Meloni, and J. J. Ramasco, “Effects of mobility and multi-seeding on the propagation of the COVID-19 in Spain,” *medRxiv*, p. 2020.05.09.20096339, Jan. 2020, doi: [10.1101/2020.05.09.20096339](https://doi.org/10.1101/2020.05.09.20096339).
- [13] G. A. Wellenius et al., “Impacts of State-Level Policies on Social Distancing in the United States Using Aggregated Mobility Data during the COVID-19 Pandemic,” *arXiv preprint arXiv:2004.10172*, 2020.
- [14] F. C. Coelho et al., “Assessing the potential impact of COVID-19 in Brazil: Mobility, Morbidity and the burden on the Health Care System,” *medRxiv*, p. 2020.03.19.20039131, Jan. 2020, doi: [10.1101/2020.03.19.20039131](https://doi.org/10.1101/2020.03.19.20039131).



- [15] A. Lasry *et al.*, “Timing of community mitigation and changes in reported COVID-19 and community mobility - four U.S. Metropolitan areas, February 26-April 1, 2020,” *MMWR. Morbidity and mortality weekly report*, vol. 69, no. 15, p. 451—457, 2020, doi: [10.15585/mmwr.mm6915e2](https://doi.org/10.15585/mmwr.mm6915e2).
- [16] C. Xiong *et al.*, “Data-Driven Modeling Reveals the Impact of Stay-at-Home Orders on Human Mobility during the COVID-19 Pandemic in the U.S.,” *arXiv e-prints*, p. arXiv:2005.00667, May 2020, Available: <https://arxiv.org/abs/2005.00667>
- [17] R. Goel and R. Sharma, “Mobility Based SIR Model For Pandemics–With Case Study Of COVID-19,” *arXiv preprint arXiv:2004.13015*, 2020.
- [18] L. Zhang *et al.*, “AN INTERACTIVE COVID-19 MOBILITY IMPACT AND SOCIAL DISTANCING ANALYSIS PLATFORM,” *medRxiv*, p. 2020.04.29.20085472, Jan. 2020, doi: [10.1101/2020.04.29.20085472](https://doi.org/10.1101/2020.04.29.20085472).
- [19] M. S. Warren and S. W. Skillman, “Mobility changes in response to COVID-19,” *arXiv preprint arXiv:2003.14228*, 2020.
- [20] A. Scala *et al.*, “Between Geography and Demography: Key Interdependencies and Exit Mechanisms for Covid-19,” Available at SSRN 3572141, 2020.
- [21] J. H. Fowler, S. J. Hill, N. Obradovich, and R. Levin, “The Effect of Stay-at-Home Orders on COVID-19 Cases and Fatalities in the United States,” *medRxiv*, 2020, doi: [10.1101/2020.04.13.20063628](https://doi.org/10.1101/2020.04.13.20063628).
- [22] S. Lai *et al.*, “Effect of non-pharmaceutical interventions to contain COVID-19 in China,” *Nature*, May 2020, doi: [10.1038/s41586-020-2293-x](https://doi.org/10.1038/s41586-020-2293-x).
- [23] M.-C. Chang, R. Kahn, Y.-A. Li, C.-S. Lee, C. O. Buckee, and H.-H. Chang, “Modeling the impact of human mobility and travel restrictions on the potential spread of SARS-CoV-2 in Taiwan,” *medRxiv*, p. 2020.04.07.20053439, Jan. 2020, doi: [10.1101/2020.04.07.20053439](https://doi.org/10.1101/2020.04.07.20053439).
- [24] J. Liu *et al.*, “Impact of meteorological factors on the COVID-19 transmission: A multi-city study in China.” *The Science of the total environment*, vol. 726, p. 138513, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138513](https://doi.org/10.1016/j.scitotenv.2020.138513).
- [25] M. Jahangiri, M. Jahangiri, and M. Najafgholipour, “The sensitivity and specificity analyses of ambient temperature and population size on the transmission rate of the novel coronavirus (COVID-19) in different provinces of Iran.” *The Science of the total environment*, vol. 728, p. 138872, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138872](https://doi.org/10.1016/j.scitotenv.2020.138872).
- [26] J. Xie and Y. Zhu, “Association between ambient temperature and COVID-19 infection in 122 cities from China.” *The Science of the total environment*, vol. 724, p. 138201, Jul. 2020, doi: [10.1016/j.scitotenv.2020.138201](https://doi.org/10.1016/j.scitotenv.2020.138201).
- [27] Y. Ma *et al.*, “Effects of temperature variation and humidity on the death of COVID-19 in Wuhan, China.” *The Science of the total environment*, vol. 724, p. 138226, Jul. 2020, doi: [10.1016/j.scitotenv.2020.138226](https://doi.org/10.1016/j.scitotenv.2020.138226).
- [28] R. Tosepu *et al.*, “Correlation between weather and Covid-19 pandemic in Jakarta, Indonesia.” *The Science of the total environment*, vol. 725, p. 138436, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138436](https://doi.org/10.1016/j.scitotenv.2020.138436).



- [29] Á. Briz-Redón and Á. Serrano-Aroca, “A spatio-temporal analysis for exploring the effect of temperature on COVID-19 early evolution in Spain.” *The Science of the total environment*, vol. 728, p. 138811, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138811](https://doi.org/10.1016/j.scitotenv.2020.138811).
- [30] N. Iqbal, Z. Fareed, F. Shahzad, X. He, U. Shahzad, and M. Lina, “The nexus between COVID-19, temperature and exchange rate in Wuhan city: New findings from partial and multiple wavelet coherence.” *The Science of the total environment*, vol. 729, p. 138916, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138916](https://doi.org/10.1016/j.scitotenv.2020.138916).
- [31] M. F. Bashir et al., “Correlation between climate indicators and COVID-19 pandemic in New York, USA.” *The Science of the total environment*, vol. 728, p. 138835, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138835](https://doi.org/10.1016/j.scitotenv.2020.138835).
- [32] C. Del Rio and A. Camacho-Ortiz, “Will environmental changes in temperature affect the course of COVID-19?” *The Brazilian journal of infectious diseases : an official publication of the Brazilian Society of Infectious Diseases*, May 2020, doi: [10.1016/j.bjid.2020.04.007](https://doi.org/10.1016/j.bjid.2020.04.007).
- [33] Y. Yao et al., “No association of COVID-19 transmission with temperature or UV radiation in Chinese cities.” *The European respiratory journal*, vol. 55, no. 5, May 2020, doi: [10.1183/13993003.00517-2020](https://doi.org/10.1183/13993003.00517-2020).
- [34] M. Ujiie, S. Tsuzuki, and N. Ohmagari, “Effect of temperature on the infectivity of COVID-19.” *International Journal of Infectious Diseases*, vol. 95, pp. 301–303, Apr. 2020, doi: [10.1016/j.ijid.2020.04.068](https://doi.org/10.1016/j.ijid.2020.04.068).
- [35] P. Mecenas, R. Bastos, A. Vallinoto, and D. Normando, “Effects of temperature and humidity on the spread of COVID-19: A systematic review.” *medRxiv*, p. 2020.04.14.20064923, Jan. 2020, doi: [10.1101/2020.04.14.20064923](https://doi.org/10.1101/2020.04.14.20064923).
- [36] A. Vantarakis, I. Chatziprodromidou, and T. Apostolou, “COVID-19 and Environmental factors. A PRISMA-compliant systematic review,” *medRxiv*, p. 2020.05.10.20069732, Jan. 2020, doi: [10.1101/2020.05.10.20069732](https://doi.org/10.1101/2020.05.10.20069732).
- [37] M. F. F. Sobral, G. B. Duarte, A. I. G. da Penha Sobral, M. L. M. Marinho, and A. de Souza Melo, “Association between climate variables and global transmission of SARS-CoV-2.” *Science of the Total Environment*, vol. 729, p. 138997, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138997](https://doi.org/10.1016/j.scitotenv.2020.138997).
- [38] H. Eslami and M. Jalili, “The role of environmental factors to transmission of SARS-CoV-2 (COVID-19).” *AMB Express*, vol. 10, no. 1, p. 92, May 2020, doi: [10.1186/s13568-020-01028-0](https://doi.org/10.1186/s13568-020-01028-0).
- [39] Y. Wu et al., “Effects of temperature and humidity on the daily new cases and new deaths of COVID-19 in 166 countries,” *Science of The Total Environment*, vol. 729, p. 139051, 2020, doi: <https://doi.org/10.1016/j.scitotenv.2020.139051>.
- [40] S. Gupta, G. S. Raghuwanshi, and A. Chanda, “Effect of weather on COVID-19 spread in the US: A prediction model for India in 2020.” *The Science of the total environment*, vol. 728, p. 138860, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138860](https://doi.org/10.1016/j.scitotenv.2020.138860).
- [41] M. Şahin, “Impact of weather on COVID-19 pandemic in Turkey.” *The Science of the total environment*, vol. 728, p. 138810, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138810](https://doi.org/10.1016/j.scitotenv.2020.138810).
- [42] P. Jüni et al., “Impact of climate and public health interventions on the COVID-19 pandemic: A prospective cohort study.” *Canadian Medical Association Journal*, May 2020, doi: [10.1503/cmaj.200920](https://doi.org/10.1503/cmaj.200920).



- [43] Y. Jiang, X.-J. Wu, and Y.-J. Guan, "Effect of ambient air pollutants and meteorological variables on COVID-19 incidence." *Infection control and hospital epidemiology*, pp. 1–11, May 2020, doi: [10.1017/ice.2020.222](https://doi.org/10.1017/ice.2020.222).
- [44] B. Pirouz, S. Shaffiee Haghshenas, B. Pirouz, S. Shaffiee Haghshenas, and P. Piro, "Development of an Assessment Method for Investigating the Impact of Climate and Urban Parameters in Confirmed Cases of COVID-19: A New Challenge in Sustainable Development." *International journal of environmental research and public health*, vol. 17, no. 8, Apr. 2020, doi: [10.3390/ijerph17082801](https://doi.org/10.3390/ijerph17082801).
- [45] A. C. Auler, F. A. M. Cássaro, V. O. da Silva, and L. F. Pires, "Evidence that high temperatures and intermediate relative humidity might favor the spread of COVID-19 in tropical climate: A case study for the most affected Brazilian cities." *The Science of the total environment*, vol. 729, p. 139090, Apr. 2020, doi: [10.1016/j.scitotenv.2020.139090](https://doi.org/10.1016/j.scitotenv.2020.139090).
- [46] M. P. Ward, S. Xiao, and Z. Zhang, "The Role of Climate During the COVID-19 epidemic in New South Wales, Australia." *Transboundary and emerging diseases*, May 2020, doi: [10.1111/tbed.13631](https://doi.org/10.1111/tbed.13631).
- [47] H. Qi *et al.*, "COVID-19 transmission in Mainland China is associated with temperature and humidity: A time-series analysis." *The Science of the total environment*, vol. 728, p. 138778, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138778](https://doi.org/10.1016/j.scitotenv.2020.138778).
- [48] D. N. Prata, W. Rodrigues, and P. H. Bermejo, "Temperature significantly changes COVID-19 transmission in (sub)tropical cities of Brazil." *The Science of the total environment*, vol. 729, p. 138862, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138862](https://doi.org/10.1016/j.scitotenv.2020.138862).
- [49] P. Shi *et al.*, "Impact of temperature on the dynamics of the COVID-19 outbreak in China." *The Science of the total environment*, vol. 728, p. 138890, Apr. 2020, doi: [10.1016/j.scitotenv.2020.138890](https://doi.org/10.1016/j.scitotenv.2020.138890).
- [50] J. Demongeot, Y. Flet-Berliac, and H. Seligmann, "Temperature Decreases Spread Parameters of the New Covid-19 Case Dynamics." *Biology*, vol. 9, no. 5, May 2020, doi: [10.3390/biology9050094](https://doi.org/10.3390/biology9050094).
- [51] M. Effenberger, A. Kronbichler, J. I. Shin, G. Mayer, H. Tilg, and P. Perco, "Association of the COVID-19 pandemic with Internet Search Volumes: A Google Trends(TM) Analysis." *International journal of infectious diseases : IJID : official publication of the International Society for Infectious Diseases*, vol. 95, pp. 192–197, Apr. 2020, doi: [10.1016/j.ijid.2020.04.033](https://doi.org/10.1016/j.ijid.2020.04.033).
- [52] Y.-H. Lin, C.-H. Liu, and Y.-C. Chiu, "Google searches for the keywords of "wash hands" predict the speed of national spread of COVID-19 outbreak among 21 countries." *Brain, behavior, and immunity*, Apr. 2020, doi: [10.1016/j.bbi.2020.04.020](https://doi.org/10.1016/j.bbi.2020.04.020).
- [53] A. Walker, C. Hopkins, and P. Surda, "Use of Google Trends to investigate loss-of-smell-related searches during the COVID-19 outbreak," *International Forum of Allergy & Rhinology*, vol. 10, no. 7, pp. 839–847, 2020, doi: [10.1002/alr.22580](https://doi.org/10.1002/alr.22580).
- [54] T. S. Higgins, A. W. Wu, D. Sharma, E. A. Illing, K. Rubel, and J. Y. Ting, "Correlations of Online Search Engine Trends With Coronavirus Disease (COVID-19) Incidence: Infodemiology Study." *JMIR public health and surveillance*, vol. 6, no. 2, p. e19702, May 2020, doi: [10.2196/19702](https://doi.org/10.2196/19702).



- [55] S. M. Ayyoubzadeh, S. M. Ayyoubzadeh, H. Zahedi, M. Ahmadi, and S. R Niakan Kalhor, “Predicting COVID-19 Incidence Through Analysis of Google Trends Data in Iran: Data Mining and Deep Learning Pilot Study.” *JMIR Public Health and Surveillance*, vol. 6, no. 2, p. e18828, Apr. 2020, doi: [10.2196/18828](https://doi.org/10.2196/18828).
- [56] X. Yuan, J. Xu, S. Hussain, H. Wang, N. Gao, and L. Zhang, “Trends and Prediction in Daily New Cases and Deaths of COVID-19 in the United States: An Internet Search-Interest Based Model.” *Exploratory research and hypothesis in medicine*, vol. 5, no. 2, pp. 1–6, Apr. 2020, doi: [10.14218/ERHM.2020.00023](https://doi.org/10.14218/ERHM.2020.00023).
- [57] A. Mavragani, “Tracking COVID-19 in Europe: Infodemiology Approach.” *JMIR public health and surveillance*, vol. 6, no. 2, p. e18941, Apr. 2020, doi: [10.2196/18941](https://doi.org/10.2196/18941).
- [58] D. Hu *et al.*, “More effective strategies are required to strengthen public awareness of COVID-19: Evidence from Google Trends.” *Journal of global health*, vol. 10, no. 1, p. 011003, Jun. 2020, doi: [10.7189/jogh.10.011003](https://doi.org/10.7189/jogh.10.011003).
- [59] Y. Ortiz-Martínez, J. E. Garcia-Robled, D. L. Vásquez-Castañeda, D. K. Bonilla-Aldana, and A. J. Rodriguez-Morales, “Can Google® trends predict COVID-19 incidence and help preparedness? The situation in Colombia.” *Travel medicine and infectious disease*, p. 101703, Apr. 2020, doi: [10.1016/j.tmaid.2020.101703](https://doi.org/10.1016/j.tmaid.2020.101703).
- [60] C. Li, L. J. Chen, X. Chen, M. Zhang, C. P. Pang, and H. Chen, “Retrospective analysis of the possibility of predicting the COVID-19 outbreak from Internet searches and social media data, China, 2020.” *Euro surveillance : bulletin European sur les maladies transmissibles = European communicable disease bulletin*, vol. 25, no. 10, Mar. 2020, doi: [10.2807/1560-7917.ES.2020.25.10.2000199](https://doi.org/10.2807/1560-7917.ES.2020.25.10.2000199).
- [61] A. I. Bento, T. Nguyen, C. Wing, F. Lozano-Rojas, Y.-Y. Ahn, and K. Simon, “Evidence from internet search data shows information-seeking responses to news of local COVID-19 cases.” *Proceedings of the National Academy of Sciences of the United States of America*, May 2020, doi: [10.1073/pnas.2005335117](https://doi.org/10.1073/pnas.2005335117).
- [62] S. Springer, L. M. Menzel, and M. Zieger, “Google Trends provides a tool to monitor population concerns and information needs during COVID-19 pandemic.” *Brain, behavior, and immunity*, Apr. 2020, doi: [10.1016/j.bbi.2020.04.073](https://doi.org/10.1016/j.bbi.2020.04.073).
- [63] W. K. Zhou, A. L. Wang, F. Xia, Y. N. Xiao, and S. Y. Tang, “Effects of media reporting on mitigating spread of COVID-19 in the early phase of the outbreak.” *Mathematical biosciences and engineering : MBE*, vol. 17, no. 3, pp. 2693–2707, Mar. 2020, doi: [10.3934/mbe.2020147](https://doi.org/10.3934/mbe.2020147).
- [64] M. Bannister-Tyrrell, A. Meyer, C. Faverjon, and A. Cameron, “Preliminary evidence that higher temperatures are associated with lower incidence of COVID-19, for cases reported globally up to 29th February 2020,” *medRxiv*, p. 2020.03.18.20036731, Jan. 2020, doi: [10.1101/2020.03.18.20036731](https://doi.org/10.1101/2020.03.18.20036731).
- [65] A. Strzelecki, “The second worldwide wave of interest in coronavirus since the COVID-19 outbreaks in South Korea, Italy and Iran: A Google Trends study,” *Brain, behavior, and immunity*, pp. S0889-1591(20)30551-1, Apr. 2020, doi: [10.1016/j.bbi.2020.04.042](https://doi.org/10.1016/j.bbi.2020.04.042).
- [66] M. Schröder, “AfD-Unterstützer sind nicht abgehängt, sondern ausländerfeindlich,” Deutsches Institut für Wirtschaftsforschung (DIW), Berlin, {SOEPpapers} on {Multidisciplinary} {Panel} {Data} {Research} 975, 2018. Available: <http://hdl.handle.net/10419/181028>



- [67] E. von Elm, G. Schreiber, and C. C. Haupt, “Methodische Anleitung für Scoping Reviews (JBI-Methodologie),” *Zeitschrift für Evidenz, Fortbildung und Qualität im Gesundheitswesen*, vol. 143, pp. 1–7, Jun. 2019, doi: [10.1016/j.zefq.2019.05.004](https://doi.org/10.1016/j.zefq.2019.05.004).
- [68] M. Wang *et al.*, “Temperature significant change COVID-19 transmission in 429 cities,” *medRxiv*, 2020, doi: [10.1101/2020.02.22.20025791](https://doi.org/10.1101/2020.02.22.20025791).
- [69] N. Dragano, C. J. Rupprecht, O. Dortmann, M. Scheider, and M. Wahrendorf, “Higher risk of COVID-19 hospitalization for unemployed: An analysis of 1,298,416 health insured individuals in Germany,” *medRxiv*, 2020, doi: [10.1101/2020.06.17.20133918](https://doi.org/10.1101/2020.06.17.20133918).
- [70] S. Dohle, T. Wingen, and M. Schreiber, “Acceptance and adoption of protective measures during the COVID-19 pandemic: The role of trust in politics and trust in science.” *OSF Preprints*, May 2020. doi: [10.31219/osf.io/w52nv](https://doi.org/10.31219/osf.io/w52nv).
- [71] S. de Lusignan *et al.*, “Risk factors for SARS-CoV-2 among patients in the Oxford Royal College of General Practitioners Research and Surveillance Centre primary care network: A cross-sectional study,” *The Lancet Infectious Diseases*, doi: [10.1016/S1473-3099\(20\)30371-6](https://doi.org/10.1016/S1473-3099(20)30371-6).
- [72] B. J. Cowling *et al.*, “Impact assessment of non-pharmaceutical interventions against coronavirus disease 2019 and influenza in Hong Kong: An observational study,” *The Lancet Public Health*, vol. 5, no. 5, pp. e279–e288, May 2020, doi: [10.1016/S2468-2667\(20\)30090-6](https://doi.org/10.1016/S2468-2667(20)30090-6).
- [73] S. Openshaw, “Ecological Fallacies and the Analysis of Areal Census Data,” *Environment and Planning A: Economy and Space*, vol. 16, no. 1, pp. 17–31, 1984, doi: [10.1068/a160017](https://doi.org/10.1068/a160017).
- [74] J. Pearl and E. Bareinboim, “External validity: From do-calculus to transportability across populations,” *Statistical Science*, vol. 29, no. 4, pp. 579–595, Nov. 2014, doi: [10.1214/14-sts486](https://doi.org/10.1214/14-sts486).
- [75] Google LLC, “Google Trends, search term "corona".” Accessed: Jun. 25, 2020. [Online]. Available: <https://www.google.com/trends>
- [76] Robert Koch-Institut (RKI), “Fallzahlen in Deutschland (COVID-19).” Available: [https://www.rki.de/DE/Content/InfAZ/N/Neuartiges\\_Coronavirus/Fallzahlen.html](https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Fallzahlen.html)
- [77] Google LLC, “Google COVID-19 community mobility reports.” Accessed: Jun. 25, 2020. [Online]. Available: <https://www.google.com/covid19/mobility/>
- [78] Deutscher Wetterdienst (DWD) Climate Data Center (CDC), “Recent daily station observations (temperature, pressure, precipitation, sunshine duration, etc.) For Germany, quality control not completed yet, version recent.” Accessed: Jun. 25, 2020. [Online]. Available: [https://opendata.dwd.de/climate\\_environment/CDC/observations\\_germany/climate/daily/kl/recent/](https://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/daily/kl/recent/)
- [79] Bundesinstitut für Bau-, Stadt- und Raumforschung (BBSR), “INKAR – Indikatoren und Karten zur Raum- und Stadtentwicklung.” Accessed: Jun. 25, 2020. [Online]. Available: <https://www.inkar.de/>
- [80] T. Mitze, R. Kosfeld, J. Rode, and K. Wälde, “Face masks considerably reduce COVID-19 cases in Germany: A synthetic control method approach,” Institute of Labor Economics (IZA), IZA Discussion Papers 13319, 2020. Available: <https://EconPapers.repec.org/RePEc:iza:izadps:dp13319>



- [81] O. Gencoglu and M. Gruber, “Causal modeling of twitter activity during COVID-19,” *medRxiv*, 2020, doi: [10.1101/2020.05.16.20103903](https://doi.org/10.1101/2020.05.16.20103903).
- [82] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman, *Causation, prediction, and search*. MIT press, 2000.
- [83] J. Pearl, *Causality*. Cambridge: Cambridge University Press, 2009. Available: <https://www.cambridge.org/core/books/causality/B0046844FAE10CBF274D4ACBDAEB5F5B>
- [84] S. Greenland, J. Pearl, and J. M. Robins, “Causal Diagrams for Epidemiologic Research,” *Epidemiology*, vol. 10, no. 1, pp. 37–48, 1999, Available: [https://journals.lww.com/epidem/Fulltext/1999/01000/Causal\\_Diagrams\\_for\\_Epidemiologic\\_Research.8.aspx](https://journals.lww.com/epidem/Fulltext/1999/01000/Causal_Diagrams_for_Epidemiologic_Research.8.aspx)
- [85] L. Henckel, E. Perković, and M. H. Maathuis, “Graphical Criteria for Efficient Total Effect Estimation via Adjustment in Causal Linear Models,” *arXiv e-prints*, p. arXiv:1907.02435, Jul. 2019, Available: <https://arxiv.org/abs/1907.02435>
- [86] R Core Team, *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing, 2019. Available: <https://www.R-project.org/>
- [87] M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann, “Causal Inference Using Graphical Models with the R Package *pcalg*,” *Journal of Statistical Software*, vol. 47, no. 11, pp. 1–26, 2012, doi: [10.18637/jss.v047.i11](https://doi.org/10.18637/jss.v047.i11).
- [88] J. M. Hilbe and W. H. Greene, “4 - Count Response Regression Models,” in *Essential Statistical Methods for Medical Statistics*, C. R. Rao, J. P. Miller, and D. C. Rao, Eds., Boston: North-Holland, 2011, pp. 104–145. Available: <http://www.sciencedirect.com/science/article/pii/B9780444537379500074>
- [89] E. Perković, J. Textor, M. Kalisch, and M. H. Maathuis, “A Complete Generalized Adjustment Criterion,” *arXiv e-prints*, p. arXiv:1507.01524, Jul. 2015, Available: <https://arxiv.org/abs/1507.01524>
- [90] W. N. Venables and B. D. Ripley, *Modern applied statistics with s*, Fourth. New York: Springer, 2002. Available: <http://www.stats.ox.ac.uk/pub/MASS4>
- [91] W. O. Kermack and A. G. McKendrick, “Contributions to the mathematical theory of epidemics–i. 1927,” *Bulletin of mathematical biology*, vol. 53, no. 1–2, p. 33—55, 1991, doi: [10.1007/bf02464423](https://doi.org/10.1007/bf02464423).
- [92] M. an der Heiden and U. Buchholz, “Modellierung von Beispielszenarien der SARS-CoV-2-Epidemie 2020 in Deutschland,” 2020, doi: [10.25646/6571.2](https://doi.org/10.25646/6571.2).
- [93] H. R. Kunsch, “The jackknife and the bootstrap for general stationary observations,” *Ann. Statist.*, vol. 17, no. 3, pp. 1217–1241, Sep. 1989, doi: [10.1214/aos/1176347265](https://doi.org/10.1214/aos/1176347265).
- [94] C. A. Field and A. H. Welsh, “Bootstrapping clustered data,” *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 69, no. 3, pp. 369–390, 2007, Available: <http://www.jstor.org/stable/4623274>



## Дополнительные интернет источники

95. <https://www.kaggle.com/code/azizozmen/heart-failure-predict-8-classification-techniques/input?select=heart.csv>
96. <https://www.kaggle.com/andrewmvd/heart-failure-clinical-data>
97. [https://pandas.pydata.org/docs/reference/api/pandas.read\\_csv.html](https://pandas.pydata.org/docs/reference/api/pandas.read_csv.html)
98. [https://www.researchgate.net/publication/49814836\\_Problematic\\_standard\\_errors\\_and\\_confidence\\_intervals\\_for\\_skewness\\_and\\_kurtosis](https://www.researchgate.net/publication/49814836_Problematic_standard_errors_and_confidence_intervals_for_skewness_and_kurtosis)
99. [https://www.researchgate.net/publication/304577646\\_Young\\_consumers'\\_intention\\_towards\\_buying\\_green\\_products\\_in\\_a\\_developing\\_nation\\_Extending\\_the\\_theory\\_of\\_planned\\_behavior](https://www.researchgate.net/publication/304577646_Young_consumers'_intention_towards_buying_green_products_in_a_developing_nation_Extending_the_theory_of_planned_behavior)
100. [https://www.researchgate.net/publication/314032599\\_TO\\_DETERMINE\\_SKEWNESS\\_MEAN\\_AND\\_DEVIATION\\_WITH\\_A\\_NEW\\_APPROACH\\_ON\\_CONTINUOUS\\_DATA](https://www.researchgate.net/publication/314032599_TO_DETERMINE_SKEWNESS_MEAN_AND_DEVIATION_WITH_A_NEW_APPROACH_ON_CONTINUOUS_DATA)
101. <https://imaging.mrc-cbu.cam.ac.uk/statswiki/FAQ/Simon>
102. [https://www.researchgate.net/publication/263372601\\_Resistance\\_motivations\\_trust\\_and\\_intention\\_to\\_use\\_mobile\\_financial\\_services](https://www.researchgate.net/publication/263372601_Resistance_motivations_trust_and_intention_to_use_mobile_financial_services)
103. <https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html>
104. <https://machinelearningmastery.com/power-transforms-with-scikit-learn/>
105. [https://en.wikipedia.org/wiki/Dummy\\_variable\\_\(statistics\)](https://en.wikipedia.org/wiki/Dummy_variable_(statistics))
106. <https://www.displayr.com/what-are-dummy-variables/>
107. <https://stattrek.com/multiple-regression/dummy-variables.aspx>
108. <https://www.statisticshowto.com/dummy-variables/>
109. [https://en.wikipedia.org/wiki/Feature\\_scaling](https://en.wikipedia.org/wiki/Feature_scaling)
110. <https://www.dataschool.io/comparing-supervised-learning-algorithms/>
111. <https://machinelearningmastery.com/handle-missing-data-python/>
112. <https://www.kaggle.com/kaanboke/the-most-used-methods-to-deal-with-missing-values>
113. <https://www.kaggle.com/karnikakapoor/fetal-health-classification>
114. <https://www.kaggle.com/karnikakapoor/heart-failure-prediction-ann>
115. <https://www.kaggle.com/kaanboke/feature-selection-the-most-common-methods-to-know>
116. <https://www.kaggle.com/kaanboke/the-most-common-evaluation-metrics-a-gentle-intro>
117. <https://www.kaggle.com/kaanboke/beginner-friendly-end-to-end-ml-project-enjoy>



## Приложение: Программный код

### EDA Блок. Создание Таблиц и графиков

```
import sys

path="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report"
pathIm="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/ASSETS/Image"
try:
    raw_df = pd.read_csv(path+'/data/heart.csv')
except:
    raw_df =pd.read_csv(path+'/data/heart.csv')

df=raw_df
he=raw_df.head(40)
print(raw_df.head(10))
des0=raw_df[raw_df['HeartDisease']==0].describe().T.agg(':{,.2f}'.format)
des1=raw_df[raw_df['HeartDisease']==1].describe().T.agg(':{,.2f}'.format)

cat = ['Sex',
'ChestPainType','FastingBS','RestingECG','ExerciseAngina',
'ST_Slope','HeartDisease']
num = ['Age','RestingBP','Cholesterol','MaxHR','Oldpeak']

import seaborn as sns
categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
numerical_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
numerical=numerical_columns

index = 0
plt.figure(figsize=(20,20))
for feature in numerical:
    if feature != "HeartDisease":
        index += 1
        plt.subplot(2, 3, index)
        sns.boxplot(x='HeartDisease', y=feature, data=df)

plt.savefig(pathIm + '/EDA1.png')

numerical_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])

# checking boxplots
def boxplots_custom(dataset, columns_list, rows, cols, suptitle):
    fig, axs = plt.subplots(rows, cols, sharey=True, figsize=(13,5))
    fig.suptitle(suptitle,y=1, size=25)
    axs = axs.flatten()
    for i, data in enumerate(columns_list):
        sns.boxplot(data=dataset[data], orient='h', ax=axs[i])
        axs[i].set_title(data + ', skewness is: ' + str(round(dataset[data].skew(axis = 0, skipna = True),2)))
    plt.tight_layout()
    plt.savefig(pathIm + '/EDA2.png')

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(11,17))
```



```
fig.suptitle('Features vs Class\n', size = 18)

sns.boxplot(ax=axes[0, 0], data=raw_df, x='Sex', y='Age',
palette='Spectral')
axes[0,0].set_title("Age distribution");

sns.boxplot(ax=axes[0,1], data=raw_df, x='Sex', y='RestingBP',
palette='Spectral')
axes[0,1].set_title("RestingBP distribution");

sns.boxplot(ax=axes[1, 0], data=raw_df, x='Sex', y='Cholesterol',
palette='Spectral')
axes[1,0].set_title("Cholesterol distribution");

sns.boxplot(ax=axes[1, 1], data=raw_df, x='Sex', y='MaxHR',
palette='Spectral')
axes[1,1].set_title("MaxHR distribution");

sns.boxplot(ax=axes[2, 0], data=raw_df, x='Sex', y='Oldpeak',
palette='Spectral')
axes[2,0].set_title("Oldpeak distribution");

sns.boxplot(ax=axes[2, 1], data=raw_df, x='Sex', y='HeartDisease',
palette='Spectral')
axes[2,1].set_title("HeartDisease distribution");

plt.tight_layout()
#fig.show()
plt.savefig(pathIm + '/EDA3.png')

numeric_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
#categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
try:
    raw_df = pd.read_csv(path+'/data/heart.csv')
except:
    raw_df = pd.read_csv(path+'/data/heart.csv')

fig = plt.figure(figsize=(15, 10))
plt.title('Kdeplot для цифровых переменных, категория: HeartDisease')
rows, cols = 2, 3
for idx, num in enumerate(numeric_columns[:30]):
    ax = fig.add_subplot(rows, cols, idx+1)
    ax.grid(alpha = 0.7, axis = "both")
    sns.kdeplot(x = num, fill = True,color ="#3386FF", linewidth=0.6,
data = raw_df[raw_df['HeartDisease']==0], label = "Healthy")
    sns.kdeplot(x = num, fill = True,color ="#EFB000", linewidth=0.6,
data = raw_df[raw_df['HeartDisease']==1] , label = "Heart Disease")
    ax.set_xlabel(num)
    ax.legend()

fig.tight_layout()
fig.show()
plt.savefig(pathIm + '/EDA4.png')

fig = plt.figure(figsize=(15, 10))
plt.title('Kdeplot для цифровых переменных, категория: Sex')
rows, cols = 2, 3
for idx, num in enumerate(numeric_columns[:30]):
    ax = fig.add_subplot(rows, cols, idx+1)
```



```
ax.grid(alpha = 0.7, axis ="both")
sns.kdeplot(x = num, fill = True,color ="#3386FF", linewidth=0.6,
data = raw_df[raw_df['Sex']=="M"], label = "M")
sns.kdeplot(x = num, fill = True,color ="#EFB000", linewidth=0.6,
data = raw_df[raw_df['Sex']=="F"], label = "F")
ax.set_xlabel(num)
ax.legend()

fig.tight_layout()
fig.show()
plt.savefig(pathIm + '/EDA5.png')

from matplotlib import *
from matplotlib.colors import ListedColormap
import matplotlib
import matplotlib.pyplot as plt
import sys
import numpy as np
import pandas as pd
import seaborn as sns
import plotly.express as px
import plotly as pplt
from matplotlib.colors import ListedColormap
import seaborn as sns
import pandas as pd
import numpy as np

raw_df = pd.read_csv(path+'/data/heart.csv')
fig = plt.figure(figsize=(25, 10))
fig = px.scatter_3d(raw_df,
                     x='RestingBP',
                     y='Age',
                     z='Sex',
                     color='HeartDisease')

fig.show()
fig.write_html(pathIm + '/Bubble3D.html')

with open("C:/IPYNBgesamt/ASNI-FEN/ASNI-
Report/ASSETS/Image/EDA6.png", 'wb') as f:
    f.write(pplt.io.to_image(fig, width=1200, height=800,
format='png'))

df = pd.read_csv(path+'/data/heart.csv')
#print(df.head())

def replace_zero_cholesterol(df):
    # Step 1: Create age groups and calculate average cholesterol
    for each group
        age_bins = [10, 20, 30, 40, 50, 60, 70, 80]
        age_labels = [f'{start}-{end}' for start, end in zip(age_bins[:-1], age_bins[1:])]
        df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins,
        labels=age_labels, right=False)
        average_cholesterol_by_age =
    df.groupby('AgeGroup')['Cholesterol'].mean()

    # Step 2: Replace zero values in 'Cholesterol' with average
    values based on age groups
    def replace_zero(row):
        if row['Cholesterol'] == 0:
            return average_cholesterol_by_age[row['AgeGroup']]
        else:
            return row['Cholesterol']
```



```
df['Cholesterol'] = df.apply(replace_zero, axis=1)

# Drop the temporary 'AgeGroup' column
df.drop(columns=['AgeGroup'], inplace=True)

# Example usage:
replace_zero_cholesterol(df)

fig = px.scatter(df, y = 'Age',x='Cholesterol', color='Cholesterol')
fig.update_layout(title=f'Bubble Chart Cholesterol')
fig.show(renderer="svg")

import plotly.io as pio
pio.write_image(fig, pathIm + '/EDA7.png', width=1200, height=800,
format='png', scale=6)

#Importing plotly and cufflinks in offline mode
import cufflinks as cf
import plotly.offline
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)

import warnings
warnings.filterwarnings('ignore')
warnings.warn("this will not show")
plt.rcParams["figure.figsize"] = (10,6)
pd.set_option('max_colwidth',200)

pd.set_option('display.max_columns', 200)
pd.set_option('display.float_format', lambda x: '%.3f' % x)

import colorama
from colorama import Fore, Style # makes strings colored
from termcolor import colored

import numpy as np
import pandas as pd
import cufflinks as cf
import matplotlib.pyplot as plt
import plotly.io as pio
cf.go_online()
cf.set_config_file(offline=False)
cf.get_config_file()
path="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report"
pathIm="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/ASSETS/Image"

raw_df = pd.read_csv(path+'/data/heart.csv')
categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
numerical_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
numerical=numerical_columns
#print(numerical)
df=raw_df[numerical].head(40)
#print(df)

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly as pl
import cufflinks as cf
import plotly.offline as po
po.init_notebook_mode(connected=True)
```



```
cf.go_offline()
df.iplot(kind='histogram', theme='white', asImage=True,
dimensions=(800,500))
fig=df.iplot(kind='histogram', theme='white', asFigure=True,
dimensions=(1200,800))
fig.write_image(pathIm + "/EDA8.png")

import seaborn as sns
import matplotlib.pyplot as plt

categorical_columns = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
numerical_columns = list(raw_df.loc[:,['Age', 'RestingBP',
'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']])
numerical=numerical_columns
#print(numerical)
df=raw_df[numerical]
#print(df.head(10))

plt.figure(figsize=(15, 8), dpi=800)
corr_matrix = df.corr()
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
heatmap = sns.heatmap(corr_matrix, annot=True, mask=mask,
cmap='viridis', linewidth=.5, annot_kws={'size': 24})
heatmap.set_title('Triangle Correlation Heatmap',
fontdict={'fontsize':24, 'weight':'bold'}, pad=16);
sns.set(font_scale=1.5)
#plt.savefig('C:\IPYNBgesamt\ASNI-FEN\ASNI-
Report\ASSETS\Image\EDA19.png')
plt.savefig(pathIm+'/EDA9.png')

#fig=plt.figure(figsize=(16, 10), dpi=600)
pathIm="C:/IPYNBgesamt/ASNI-FEN/ASNI-Report/ASSETS/Image"
raw_df[numerical].iplot(kind='histogram', subplots=True,bins=50,
theme='white',asImage=True,dimensions=(1200,800))
fig=raw_df[numerical].iplot(kind='histogram', subplots=True,bins=50,
theme='white',asFigure=True,dimensions=(1200,800))
fig.write_image(pathIm + "/EDA10.png")
fig.show()

import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
cat = ['Sex', 'ChestPainType','FastingBS','RestingECG',
'ExerciseAngina',
'ST_Slope','HeartDisease']
num = ['Age','RestingBP','Cholesterol','MaxHR','Oldpeak']

df=raw_df

# Using facet_row and or facet_col arguments to create Sub plots
fig = px.scatter(df,
                  x=df.Age,
                  y=df.Cholesterol,
                  color=df.HeartDisease,
                  facet_col=df.FastingBS,
                  facet_row=df.Sex,
                  color_discrete_map={1: "#FF5722",0: "#7CB342"},
                  width=950,
                  height=800,
                  title="HeartDisease Data")

fig.update_layout(
    plot_bgcolor= "#dcedc1",
```



```
        paper_bgcolor="#FFFDE7",
    )
fig.show()
fig.write_image(pathIm + '/EDA11.png', scale=4)

colors = px.colors.cyclical.Twilight

HDValues={
    0:'Healthy',
    1:'Heart Disease'
}
#df = raw_df.HeartDisease.replace(HDValues)
df = raw_df
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
                    '#ffffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale =
0.55)
fig, ax = plt.subplots(4, 2, figsize = (6.5, 7.5))
for indx, (column, axes) in list(enumerate(list(zip(cat,
ax.flatten())))):

    sns.violinplot(ax = axes, x = df[column],
                    y = df['Age'],
                    scale = 'width', linewidth = 0.5,
                    palette = colors, inner = None)

    plt.setp(axes.collections, alpha = 0.3)

    sns.stripplot(ax = axes, x = df[column],
                  y = df['Age'],
                  palette = colors, alpha = 0.9,
                  s = 1.5, jitter = 0.07)
    sns.pointplot(ax = axes, x = df[column],
                  y = df['Age'],
                  color = '#ff5736', scale = 0.25,
                  estimator = np.mean, ci = 'sd',
                  errwidth = 0.5, capsizer = 0.15, join = True)

    plt.setp(axes.lines, zorder = 100)
    plt.setp(axes.collections, zorder = 100)

else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]

plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA12.png')

sns.set_theme(rc = {'figure.dpi': 120, 'axes.labelsize': 8,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
                    '#ffffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale =
0.65)

fig, ax = plt.subplots(5, 1, figsize = (10, 10))

for indx, (column, axes) in list(enumerate(list(zip(num,
ax.flatten())))):

    sns.scatterplot(ax = axes, y = df[column].index, x = df[column],
                    hue = df['HeartDisease'], palette = 'magma',
                    alpha = 0.8)

else:
```



```
[axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]

plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA13.png')

sns.set_theme(rc = {'figure.dpi': 120, 'axes.labelsize': 8,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
                    '#ffffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale =
0.65)

fig, ax = plt.subplots(5, 1, figsize = (10, 14))

for indx, (column, axes) in list(enumerate(list(zip(num,
ax.flatten())))):

    sns.histplot(ax = axes, x = df[column], hue =
df['HeartDisease'],
                  palette = 'magma', alpha = 0.8, multiple = 'stack')

    legend = axes.get_legend() # sns.histplot has some issues with
legend
    handles = legend.legendHandles
    legend.remove()
    axes.legend(handles, ['0', '1'], title = 'HeartDisease', loc =
'upper right')
    Quantiles = np.quantile(df[column], [0, 0.25, 0.50, 0.75, 1])

    for q in Quantiles: axes.axvline(x = q, linewidth = 0.5, color =
'r')

plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA14.png')

df2 = df.groupby('Sex').agg({'Age' : 'mean',
                            "ChestPainType":'count','RestingBP':'mean','Cholesterol':'mean',
                            'FastingBS':'sum','RestingECG':'count','MaxHR':'mean','ExerciseAngin
a':'count','Oldpeak':'mean',
                            'ST_Slope':'count','HeartDisease':'sum'})
df2
fig=px.bar(data_frame=df2, barmode='group',
            title = "<b>Gender wise
Analyzing</b>",template="plotly_dark")
fig.show()
fig.write_image(pathIm + '/EDA15.png',scale=4)

try:
    heart_dft = pd.read_csv(path+'/data/heart.csv')
except:
    heart_dft=pd.read_csv(path+'/data/heart.csv')

#print(heart_dft.head())

sex_color = dict({"Male": "#2986cc", "Female": "#c90076"})
plt.style.use("fivethirtyeight")
heart_dft["Sex"] = heart_dft["Sex"].map({ "M": "Male", "F":
"Female"})
heart_dft["Sex"]
#print(heart_dft["Sex"].head())
heart_dft["HeartDisease"] = heart_dft["HeartDisease"].map({0: "No",
1: "Yes"})
```



```
#heart_dft.info()

filtheart_dft = heart_dft["Cholesterol"] > 0
heart_dft_chol_n0 = heart_dft[filtheart_dft]
#heart_dft_chol_n0.head()

#heart_dft=heart
print(heart_dft.head())
sex_color = dict({"Male": "#2986cc", "Female": "#c90076"})
plt.style.use("fivethirtyeight")
heart_dft["Sex"] = heart_dft["Sex"].map({"M": "Male", "F": "Female"})
heart_dft["Sex"]
heart_dft["HeartDisease"] = heart_dft["HeartDisease"].map({0: "No", 1: "Yes"})
filtheart_dft = heart_dft["Cholesterol"] > 0
heart_dft_chol_n0 = heart_dft[filtheart_dft]

g=sns.JointGrid(
    data=heart_dft, x="Age", y="Cholesterol", hue="Sex",
    palette=sex_color
).plot(sns.scatterplot, sns.histplot)

plt.legend(title='Company', fontsize=20)
plt.xlabel('Age', fontsize=10);
plt.ylabel('Cholesterol', fontsize=10);
plt.title('Sales Data', fontsize=12)
plt.tick_params(axis='both', which='major', labelsize=10)
#plt.show()
g.savefig(pathIm + '/EDA16.png')

import seaborn as sns
import patchworklib as pw
sns.set_theme()
pw.overwrite_axisgrid()
#heart=raw_df
#raw_df = heart
iris = sns.load_dataset("iris")
tips = sns.load_dataset("tips")

# An lmplot
g0 = sns.lmplot(x="total_bill", y="tip", hue="smoker", data=tips,
                 palette=dict(Yes="g", No="m"))
g0 = pw.load_seaborngrid(g0, label="g0")

# A Pairplot
g1 = sns.pairplot(iris, hue="species")
g1 = pw.load_seaborngrid(g1, label="g1")

# A relplot
g2 = sns.relplot(data=tips, x="total_bill", y="tip", col="time",
                  hue="time",
                  size="size", style="sex", palette=["b", "r"],
                  sizes=(10, 100))
g2 = pw.load_seaborngrid(g2, label="g2")

# A JointGrid
#g3 = sns.jointplot(x='Cholesterol',y='FastingBS',data=raw_df,
#hue="Sex")
g3 = sns.jointplot(x='Cholesterol',y='Age',data=raw_df, hue="Sex")
g3 = pw.load_seaborngrid(g3, label="g3")
((g0|g1)|g3)|g2).savefig(pathIm + '/EDA17.png')

try:
    heart_dft = pd.read_csv(path+'data/heart.csv')
```



```
except:  
    heart_dft=pd.read_csv(path+'/data/heart.csv')  
  
#print(heart_dft.head())  
  
sex_color = dict({"Male": "#2986cc", "Female": "#c90076"})  
plt.style.use("fivethirtyeight")  
heart_dft["Sex"] = heart_dft["Sex"].map({"M": "Male", "F": "Female"})  
heart_dft["Sex"]  
#print(heart_dft["Sex"].head())  
heart_dft["HeartDisease"] = heart_dft["HeartDisease"].map({0: "No",  
1: "Yes"})  
#heart_dft.info()  
  
filtheart_dft = heart_dft["Cholesterol"] > 0  
heart_dft_chol_n0 = heart_dft[filtheart_dft]  
  
Chol_mean_f = (  
    heart_dft_chol_n0[["Sex", "Cholesterol"]]  
    .groupby(["Sex"])  
    .mean("Cholesterol")  
    .loc["Female", "Cholesterol"]  
).round()  
  
Chol_mean_m = (  
    heart_dft_chol_n0[["Sex", "Cholesterol"]]  
    .groupby(["Sex"])  
    .mean("Cholesterol")  
    .loc["Male", "Cholesterol"]  
).round()  
  
print("for Female =", Chol_mean_f, "for Male =", Chol_mean_m)  
  
plt.figure(figsize=(10, 5))  
sns.set_context("paper")  
  
kdeplt = sns.kdeplot(  
    data=heart_dft_chol_n0,  
    x="Cholesterol",  
    hue="Sex",  
    palette=sex_color,  
    alpha=0.7,  
    lw=2,  
)  
  
kdeplt.set_title("Cholesterol values distribution\n Male VS Female",  
    fontsize=12)  
kdeplt.set_xlabel("Cholesterol", fontsize=12)  
plt.axvline(x=Chol_mean_f, color="#c90076", ls="--", lw=1.3)  
plt.axvline(x=Chol_mean_m, color="#2986cc", ls="--", lw=1.3)  
plt.text(108, 0.00612, "Mean Cholesterol / Male", fontsize=10,  
color="#2986cc")  
plt.text(260, 0.006, "Mean Cholesterol / Female", fontsize=10,  
color="#c90076")  
kdeplt.figure.savefig(pathIm + '/EDA18.png')  
  
#Kategorial  
from matplotlib import *  
from matplotlib.colors import ListedColormap  
import matplotlib  
import sys  
import numpy as np  
import pandas as pd  
import seaborn as sns
```



```
import plotly.express as px
from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap
import seaborn as sns
import pandas as pd
import numpy as np
pio.renderers

def auto_fmt (pct_value):
    return
' {:.0f}\n({:.1f}%)'.format(raw_df['HeartDisease'].value_counts().sum()
() * pct_value/100, pct_value)

try:
    raw_df = pd.read_csv(path+'/data/heart.csv')
except:
    raw_df = pd.read_csv(path+'/data/heart.csv')
HDValues={
    0:'Healthy',
    1:'Heart Disease'
}

df = raw_df.HeartDisease.replace(HDValues)
#print(df.head())

fig=plt.figure(figsize=(6, 6))
matplotlib.rcParams.update({'font.size': 15})

df.value_counts().plot.pie(explode=[0.1, 0.1],
                           autopct=auto_fmt,
                           textprops={'fontsize': 16},
                           shadow=True)

plt.title('Healthy vs Heart Disease', color='Red', pad=15,
fontsize=20);
plt.axis('off');
fig.show()
plt.savefig(pathIm + '/EDA31.png')

plt.figure(figsize=(6, 6))
matplotlib.rcParams.update({'font.size': 15})

raw_df.Sex.value_counts().plot.pie(explode=[0.1, 0.1],
#                           autopct='%.1.2f%%',
                           autopct=auto_fmt,
                           textprops={'fontsize': 16},
                           shadow=True)
plt.title('Sex', color='Red', pad=10, fontsize=20);
plt.axis('off');

fig.show()
plt.savefig(pathIm + '/EDA32.png')

fig, ax = plt.subplots (3, 2, figsize=(16, 16))
ax_rst = []

for i in range(len(categorical_columns)):
    axs = sns.countplot(data=raw_df, x
=raw_df[categorical_columns[i]], ax=ax[int(i/2),i % 2])
    ax_rst.append(axs)
    total = raw_df[categorical_columns[i]].value_counts().sum()
    for p in axs.patches:
        value_pct = ' {:.0f}\n({:.1f}%)'.format(p.get_height(), 100 *
p.get_height()/total)
```



```
x = p.get_x() + p.get_width()/2
y = p.get_height()
axs.annotate(value_pct, (x, y), ha='center')
plt.savefig(pathIm + '/EDA33.png')

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly as pl
import cufflinks as cf
import plotly.offline as po
po.init_notebook_mode(connected=True)
cf.go_offline()

categorical = list(raw_df.loc[:,['Sex', 'ChestPainType',
'FastingBS', 'RestingECG', 'ExerciseAngina', 'ST_Slope']])
raw_df[categorical].iplot(kind='box', subplots=True,bins=50,
theme='white', asImage=True, dimensions=(800,500))
fig=raw_df[categorical].iplot(kind='box', subplots=True, bins=50,
theme='white', asFigure=True, dimensions=(800,500))
fig.write_image(pathIm + "/EDA34.png")

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

raw_df = pd.read_csv(path+"/data/heart.csv")
raw_df.head()

df1=raw_df[categorical]
df1.iplot(kind='hist',
theme='white',asImage=True,dimensions=(1200,800))
fig=df1.iplot(kind='hist', theme='white',asFigure=True,
dimensions=(1200,800))
fig.write_image(pathIm + "/EDA35.png")

import plotly.express as px
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
fig=px.pie(raw_df,values='HeartDisease',names='ChestPainType',
template='plotly_dark',color_discrete_sequence=px.colors.sequential.RdBu,
title='The effect of the type of chest pain on the
disease')
fig.update_traces(textposition='inside',textinfo='percent+label')
fig.show()
fig.update_layout(width=1000, height=800)
fig.write_image(pathIm + '/EDA36.png', scale=4)

fig=px.pie(raw_df,values='HeartDisease',names='ST_Slope',hole=.4,tem
plate='plotly_dark',title='The effect of the the slope of the peak
exercise on the disease')
fig.update_traces(textposition='inside',textinfo='percent+label')
fig.update_layout(annotations=[dict(text='ST slope', x=0.5, y=0.5,
font_size=20, showarrow=False)])
fig.show()
fig.update_layout(width=1000, height=1000)
fig.write_image(pathIm + '/EDA37.png', scale=4)

df=raw_df
colors = px.colors.cyclical.Twilight
fig = make_subplots(rows=1,cols=2,
```



```
        subplot_titles=('Countplot',
                         'Percentages'),
        specs=[[{"type": "xy"}, {"type": 'domain'}]]))

fig.add_trace(go.Bar(y = df['Sex'].value_counts().values.tolist(),
                      x = df['Sex'].value_counts().index,
                      text=df['Sex'].value_counts().values.tolist(),
                      textfont=dict(size=15),
                      textposition = 'outside',
                      showlegend=False,
                      marker = dict(color = colors,
                                    line_color = 'black',
                                    line_width=3)),row = 1,col = 1)
fig.add_trace((go.Pie(labels=df['Sex'].value_counts().keys(),
                      values=df['Sex'].value_counts().values, textfont = dict(size = 16),
                      hole = .4,
                      marker=dict(colors=colors),
                      textinfo='label+percent',
                      hoverinfo='label')), row = 1, col = 2)
fig.update_yaxes(range=[0,800])
#Changing plot & figure background
fig.update_layout(
    paper_bgcolor= '#FFFDE7',
    plot_bgcolor= '#FFFDE7',
    title=dict(text = "Gender
Distribution",x=0.5,y=0.95),
    title_font_size=30
)
iplot(fig)
fig.write_image(pathIm + '/EDA38.png',scale=4

colors = px.colors.cyclical.Twilight
fig = make_subplots(rows=1,cols=2,
                     subplot_titles=('Countplot',
                                     'Percentages'),
                     specs=[[{"type": "xy"}, {"type": 'domain'}]]))

fig.add_trace(go.Bar(y =
df['HeartDisease'].value_counts().values.tolist(),
                      x = df['HeartDisease'].value_counts().index,
                      text=df['HeartDisease'].value_counts().values.tolist(),
                      textfont=dict(size=15),
                      textposition = 'outside',
                      showlegend=False,
                      marker = dict(color = colors,
                                    line_color = 'black',
                                    line_width=3)),row = 1,col = 1)
fig.add_trace((go.Pie(labels=df['HeartDisease'].value_counts().keys(),
                      values=df['HeartDisease'].value_counts().values, textfont = dict(size = 16),
                      hole = .4,
                      marker=dict(colors=colors),
                      textinfo='label+percent',
                      hoverinfo='label')), row = 1, col = 2)
fig.update_yaxes(range=[0,550])
#Changing plot & figure background
fig.update_layout(
    paper_bgcolor= '#FFFDE7',
    plot_bgcolor= '#FFFDE7',
```



```
        title=dict(text = "HeartDisease
Distribution",x=0.5,y=0.95),
        title_font_size=30
    )
iplot(fig)
fig.write_image(pathIm + '/EDA39.png',scale=4)

cat = ['Sex', 'ChestPainType', 'FastingBS', 'RestingECG',
       'ExerciseAngina',
       'ST_Slope', 'HeartDisease']
num = ['Age','RestingBP','Cholesterol','MaxHR','Oldpeak']

import seaborn as sns
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
                    '#ffffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale =
0.55)
fig, ax = plt.subplots(3, 2, figsize = (6.5, 9))
for indx, (column, axes) in list(enumerate(list(zip(cat,
ax.flatten())))):
    if column not in 'HeartDisease':
        sns.countplot(ax = axes, x = df[column], hue =
df['HeartDisease'], palette = colors, alpha = 1)
    else:
        [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]
    axes.legend = ax.flatten()
    axes.legend[1].legend(title = 'HeartDisease', loc = 'upper right')
    axes.legend[2].legend(title = 'HeartDisease', loc = 'upper right')
plt.show()
fig.savefig(pathIm + '/EDA40.png')

import seaborn as sns
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
                    '#ffffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale =
0.55)
fig, ax = plt.subplots(3, 2, figsize = (6.5, 9))
for indx, (column, axes) in list(enumerate(list(zip(cat[1:],
ax.flatten())))):
    sns.countplot(ax = axes, x = df[column], hue = df['Sex'],
palette = colors, alpha = 1)
    else:
        [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]
    axes.legend = ax.flatten()
    axes.legend[1].legend(title = 'Sex', loc = 'upper right')
    axes.legend[2].legend(title = 'Sex', loc = 'upper right')
plt.show()
fig.savefig(pathIm + '/EDA41.png')

import seaborn as sns
sns.set_theme(rc = {'figure.dpi': 250, 'axes.labelsize': 7,
                    'axes.facecolor': '#FFFDE7', 'grid.color':
                    '#ffffdfa',
                    'figure.facecolor': '#FFFDE7'}, font_scale =
0.55)
fig, ax = plt.subplots(3, 2, figsize = (6.5, 9))
cat2 = []
for i in cat:
    if i not in 'ChestPainType':
        cat2.append(i)
for indx, (column, axes) in list(enumerate(list(zip(cat2,
ax.flatten())))):
```



```
sns.countplot(ax = axes, x = df[column], hue =
df['ChestPainType'], palette = colors, alpha = 1)
else:
    [axes.set_visible(False) for axes in ax.flatten()[indx + 1:]]
axes_legend = ax.flatten()
axes_legend[1].legend(title = 'ChestPainType', loc = 'upper right')
axes_legend[2].legend(title = 'ChestPainType', loc = 'upper right')
plt.show()
fig.savefig(pathIm + '/EDA42.png')

ax = plt.subplot(1,2,1)
ax = sns.countplot(x='Sex', data=raw_df)
ax.bar_label(ax.containers[0])
ax =plt.subplot(1,2,2)
ax=raw_df['Sex'].value_counts().plot.pie(explode=[0.1,
0.1], autopct='%.2f%%', shadow=True);
ax.set_title(label = "Sex", fontsize = 16)
#,color='Red',font='Lucida Calligraphy')
plt.savefig(pathIm + '/EDA43.png')

heart=raw_df
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='ChestPainType', data=heart)
ax.bar_label(ax.containers[0])
plt.title("ChestPainType", fontsize=14)
ax =plt.subplot(1,2,2)
ax=heart['ChestPainType'].value_counts().plot.pie(explode=[0.1,
0.1,0.1], autopct='%.2f%%', shadow=True);
ax.set_title(label = "ChestPainType", fontsize =
20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA44.png')

ax = plt.subplot(1,2,1)
ax = sns.countplot(x='RestingECG', data=heart)
ax.bar_label(ax.containers[0])
plt.title("RestingECG", fontsize=14)

ax =plt.subplot(1,2,2)
ax=heart['RestingECG'].value_counts().plot.pie(explode=[0.1,
0.1,0.1], autopct='%.2f%%', shadow=True);
ax.set_title(label = "RestingECG", fontsize =
20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA45.png')

ax = plt.subplot(1,2,1)
ax = sns.countplot(x='ExerciseAngina', data=heart)
ax.bar_label(ax.containers[0])
plt.title("ExerciseAngina", fontsize=14)

ax =plt.subplot(1,2,2)
ax=heart['ExerciseAngina'].value_counts().plot.pie(explode=[0.1,
0.1], autopct='%.2f%%', shadow=True);
ax.set_title(label = "ExerciseAngina", fontsize =
20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA46.png')

ax = plt.subplot(1,2,1)
ax = sns.countplot(x='ST_Slope', data=heart)
ax.bar_label(ax.containers[0])
plt.title("ST_Slope", fontsize=14)

ax =plt.subplot(1,2,2)
ax=heart['ST_Slope'].value_counts().plot.pie(explode=[0.1,
0.1,0.1], autopct='%.2f%%', shadow=True);
```



```
ax.set_title(label = "ST_Slope", fontsize = 20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA47.png')

#heart=pd.read_csv('data/heart.csv')
sns.set(font_scale=1.1)
heart["Cholesterol_Category"] = pd.cut(heart["Cholesterol"] ,bins=[0, 200, 230 , 500] ,labels=["Normal","Borderline","High" ] )
print("Value Counts
:\n\n",heart['Cholesterol_Category'].value_counts())

heart.head()
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='Cholesterol_Category', data=heart)
ax.bar_label(ax.containers[0])
plt.title("Cholesterol_Categoryy", fontsize=14)

ax =plt.subplot(1,2,2)
ax=heart['Cholesterol_Category'].value_counts().plot.pie(explode=[0.1, 0.1,0.1],autopct='%.1.2f%%',shadow=True);
ax.set_title(label = "Cholesterol_Category", fontsize = 20,color='Red',font='Lucida Calligraphy');
plt.savefig(pathIm + '/EDA48.png')

heart["RestingBP_Category"] = pd.cut(heart["RestingBP"] ,bins=[0,120, 129 , 139,200]
,labels=["Normal_BP","Elevated_BP","Hypertension_Stage_1",
"Hypertension_Stage_2" ] )
print("Value Counts
:\n\n",heart['RestingBP_Category'].value_counts())
heart.sample(5)
heart['RestingBP_Category'] =
heart['RestingBP_Category'].astype(object)

plt.rcParams['legend.fontsize'] = 7
sns.set(font_scale=1.0)
ax = plt.subplot(1,2,1)
ax = sns.countplot(x='RestingBP_Category', data=heart)
ax.bar_label(ax.containers[0])
plt.axis('off');

ax =plt.subplot(1,2,2)
ax=heart['RestingBP_Category'].value_counts().plot.pie(explode=[0.1, 0.1,0.1],autopct='%.1.2f%%',shadow=True);
plt.axis('off');
plt.savefig(pathIm + '/EDA49.png')

df = heart # pd.read_csv('data/heart.csv')
male_df = df[df['Sex'] == 'M']
female_df = df[df['Sex'] == 'F']

## Grouping Datasets
male_cp_fbs = male_df.groupby(['ChestPainType',
'FastingBS']).size().reset_index().rename(columns={0: 'count'})
female_cp_fbs = female_df.groupby(['ChestPainType',
'FastingBS']).size().reset_index().rename(columns={0: 'count'})

male_st_ecg = male_df.groupby(['ST_Slope',
'RestingECG']).size().reset_index().rename(columns={0: 'count'})
female_st_ecg = female_df.groupby(['ST_Slope',
'RestingECG']).size().reset_index().rename(columns={0: 'count'})

male_ea_cp = male_df.groupby(['ExerciseAngina',
'ChestPainType']).size().reset_index().rename(columns={0: 'count'})
```



## Автоматизация Система Научных Исследований в медицине и здравоохранении «АСНИ-МЕД»

```
female_ea_cp = female_df.groupby(['ExerciseAngina',
'ChestPainType']).size().reset_index().rename(columns={0: 'count'})\n\n## Creating Sunburst Figures\nsb1 = px.sunburst(male_cp_fbs, values='count',
path=['ChestPainType', 'FastingBS'])\nsb2 = px.sunburst(female_cp_fbs, values='count',
path=['ChestPainType', 'FastingBS'])\n\nsb3 = px.sunburst(male_st_ecg, values='count', path=['ST_Slope',
'RestingECG'])\nsb4 = px.sunburst(female_st_ecg, values='count', path=['ST_Slope',
'RestingECG'])\n\nsb5 = px.sunburst(male_ea_cp, values='count',
path=['ExerciseAngina', 'ChestPainType'])\nsb6 = px.sunburst(female_ea_cp, values='count',
path=['ExerciseAngina', 'ChestPainType'])\n\n## Subplots\nfig = make_subplots(rows=3, cols=2, specs=[\n    [{"type": "sunburst"}, {"type": "sunburst"}],\n    [{"type": "sunburst"}, {"type": "sunburst"}],\n    [{"type": "sunburst"}, {"type": "sunburst"}]\n], subplot_titles=("Male Chest Pain with Fasting Blood Sugar",
"Female Chest Pain with Fasting Blood Sugar",
"Male ST Slope with Resting ECG", "Female ST Slope with Resting ECG",
"Male Exercise Angina with Chest Pain Type",
"Female Exercise Angina with Chest Pain Type"))\n\n## Plotting Figures\nfig.add_trace(sb1.data[0], row=1, col=1)\nfig.add_trace(sb2.data[0], row=1, col=2)\nfig.add_trace(sb3.data[0], row=2, col=1)\nfig.add_trace(sb4.data[0], row=2, col=2)\nfig.add_trace(sb5.data[0], row=3, col=1)\nfig.add_trace(sb6.data[0], row=3, col=2)\n\nfig.update_traces(textinfo="label+percent parent")\n\n# Update title and height\nfig.update_layout(title_text="Male vs Female Sunburst", title_x=0.5,
height=1200, width=1200, template='plotly_dark',
showlegend=False,
font=dict(
    family="Rubik",
    size=14)
)\n\nfig.show()\nfig.write_image(pathIm + '/EDA50.png', scale=6)\n\nheart_dft= heart\n#print(heart_dft.head())\nRestingECG_vs_Sex = (\n    heart_dft[["RestingECG", "Sex"]]\n    .value_counts(normalize=True)\n    .reset_index(name="Pct")\n    .sort_values(by="RestingECG")\n)\nRestingECG_vs_Sex["Pct"] = RestingECG_vs_Sex["Pct"].round(2) * 100\nRestingECG_vs_Sex.sort_values(by="Pct", ascending=False)\n\nChestPainType_vs_Sex = (
```



```
heart_dft[["ChestPainType", "Sex"]]
    .value_counts(normalize=True)
    .reset_index(name="Pct")
    .sort_values(by="ChestPainType")
)
ChestPainType_vs_Sex["Pct"] = ChestPainType_vs_Sex["Pct"].round(2) *
100
#ChestPainType_vs_Sex
plt.style.use("fivethirtyeight")
fig, ax = plt.subplots(1, 2, figsize=(12, 5), sharey=True)
palette4 = {"ASY": "#1b85b8", "ATA": "#5a5255", "NAP": "#559e83",
"TA": "#ae5a41"}
palette5 = {"LVH": "#2dc937", "Normal": "#e7b416", "ST": "#cc3232"}

sns.barplot(
    data=ChestPainType_vs_Sex,
    x="Sex",
    hue="ChestPainType",
    #errorbar=None,
    y="Pct",
    palette=palette4,
    linewidth=0.5,
    edgecolor="black",
    alpha=0.8,
    ax=ax[0],
)
for ax1 in [ax[0]]:
    for container in ax1.containers:
        values2 = container.datavals
        labels = ["{:g}%".format(val) for val in values2]
        ax1.bar_label(container, labels=labels)

ax[0].set_ylabel("Percent")
ax[0].set_xlabel("")
ax[0].set_title(
    "Regardless of the proportion of Males and Females,\n Men have
high ASY compared with Women, and the pattern is different.",
    fontsize=10,
)
sns.barplot(
    data=RestingECG_vs_Sex,
    x="Sex",
    hue="RestingECG",
    #errorbar=None,
    y="Pct",
    palette=palette5,
    linewidth=0.5,
    edgecolor="black",
    alpha=0.8,
    ax=ax[1],
)
for ax2 in [ax[1]]:
    for container in ax2.containers:
        values3 = container.datavals
        labels = ["{:g}%".format(val) for val in values3]
        ax2.bar_label(container, labels=labels)

ax[1].set_ylabel("")
ax[1].set_xlabel("")
ax[1].set_title("Men and Women have somehow same pattern of
RestingECG", fontsize=10)
```



```
plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA51.png')

ExerciseAngina_vs_Sex = (
    heart_dft[["ExerciseAngina", "Sex"]]
    .value_counts(normalize=True)
    .reset_index(name="Pct")
    .sort_values(by="ExerciseAngina")
)
ExerciseAngina_vs_Sex["Pct"] = ExerciseAngina_vs_Sex["Pct"].round(2) * 100

ST_Slope_vs_Sex = (
    heart_dft[["ST_Slope", "Sex"]]
    .value_counts(normalize=True)
    .reset_index(name="Pct")
    .sort_values(by="ST_Slope")
)
ST_Slope_vs_Sex["Pct"] = ST_Slope_vs_Sex["Pct"].round(2) * 100

plt.style.use("fivethirtyeight")
fig, ax = plt.subplots(1, 2, figsize=(12, 5), sharey=True)

palette6 = {
    "Y": "#000000",
    "N": "#009900",
}

palette7 = {"Down": "#b2d8d8", "Flat": "#66b2b2", "Up": "#004c4c"}

sns.barplot(
    data=ExerciseAngina_vs_Sex,
    x="Sex",
    hue="ExerciseAngina",
    #errorbar=None,
    y="Pct",
    palette=palette6,
    linewidth=0.5,
    edgecolor="black",
    alpha=0.8,
    ax=ax[0],
)
for ax3 in [ax[0]]:
    for container in ax3.containers:
        values2 = container.datavalues
        labels = ["{:g}%".format(val) for val in values2]
        ax3.bar_label(container, labels=labels)

ax[0].set_ylabel("Percent")
ax[0].set_xlabel("")
ax[0].set_title(
    "Almost a similar pattern between Men and Women.\n(ExerciseAngina)", fontsize=10
)

sns.barplot(
    data=ST_Slope_vs_Sex,
    x="Sex",
    hue="ST_Slope",
    #errorbar=None,
```



```
y="Pct",
palette=palette7,
linewidth=0.5,
edgecolor="black",
alpha=0.8,
ax=ax[1],
)

for ax4 in [ax[1]]:
    for container in ax4.containers:
        values3 = container.datavalues
        labels = ["{:g}%".format(val) for val in values3]
        ax4.bar_label(container, labels=labels)

ax[1].set_ylabel("")
ax[1].set_xlabel("")
ax[1].set_title(
    "A different pattern between Men and Women (ExerciseAngina)",
    fontsize=10
)

plt.tight_layout()
plt.show()
fig.savefig(pathIm + '/EDA52.png')
```



## EDA Блок. Создание промежуточного отчета

```
#System Generell
import os

today = datetime.date.today()
year = today.year
cwd = os.getcwd()
template=os.path.join(cwd, "Templates")
EDA_JSON = os.path.join(cwd, "Templates\EDA.json")

IMAGE=os.path.join(cwd, "ASSETS\Image")
InEDA_File = os.path.join(cwd, "Templates\TemplateTOC1.docx")
InEDA = os.path.join(cwd, "data\InEDA.docx")
ReportEDA = os.path.join(cwd, "OUTPUT-RESSOURCE\ReportEDA.docx")
ReportEDAr = os.path.join(cwd, "data\Kap3.docx")
ReportAll = os.path.join(cwd, "OUTPUT-
RESSOURCE\ReportReportAll.docx")
excel_filename=os.path.join(cwd, "OUTPUT/EDA-Report5.xlsx")

now=datetime.datetime.now()
timestart = now.replace(microsecond=0)
print("Programm Start: ", timestart)
timeend = datetime.datetime.now()
date_time = timeend.strftime("%Y%m%d")
print("Date:",date_time)

print("ReportEDA: ", ReportEDA)
print("InEDA_File: ", InEDA_File)
print("InEDA: ", InEDA)
print("EDA_JSON: ", EDA_JSON)
print("IMAGE: ", IMAGE)
print("ReportAll: ", ReportAll)
print("ReportEDAr: ", ReportEDAr)
print("excel_filename: ", excel_filename)

#Def Section
def getConfigFile(config):
    with open(config, encoding='utf-8') as json_file:
        return json.load(json_file)

def get_para_data(output_doc_name, paragraph):
    output_para = output_doc_name.add_paragraph()
    for run in paragraph.runs:
        output_run = output_para.add_run(run.text)
        # Run's bold data
        output_run.bold = run.bold
        # Run's italic data
        output_run.italic = run.italic
        # Run's underline data
        output_run.underline = run.underline
        # Run's color data
        output_run.font.color.rgb = run.font.color.rgb
        # Run's font data
        output_run.style.name = run.style.name
        # Paragraph's alignment data

        output_para.paragraph_format.alignment =
paragraph.paragraph_format.alignment
        output_para.paragraph_format.first_line_indent = Inches(0.25)
        output_para.paragraph_format.space_before = Pt(0)
        output_para.paragraph_format.space_after = Pt(0)
    return output_para
```



```
def replace_copy(txt):
    finder = wordapp.Selection.Find
    finder.Text = txt
    finder.Execute()
    wordapp.Selection.MoveStart

def set_repeat_table_header(row):
    tr = row._tr
    trPr = tr.get_or_add_trPr()
    tblHeader = OxmlElement('w:tblHeader')
    tblHeader.set(qn('w:val'), "true")
    trPr.append(tblHeader)
    return row

def change_table_cell(cell, background_color=None, font_color=None,
font_size=None, bold=None, italic=None):
    """ changes the background_color or font_color or font style
    (bold, italic) of this cell.
    Leave the params as 'None' if you do not want to change them.
    params:
        cell: the cell to manipulate
        background_color: name for the color, e.g. "red" or "ff0000"
        font_color:
        font_size: size in pt (e.g. 10)
        bold: requested font style. True or False, or None if it
shall remain unchanged
        italic: requested font style. True or False, or None if it
shall remain unchanged
        background_color: the color of cells background"""
    if background_color:
        shading_elm = parse_xml(r'<w:shd {}'
w:fill="{}"/>'.format(nsdecls('w'), background_color))
        cell._tc.get_or_add_tcPr().append(shading_elm)

    if font_color:
        for p in cell.paragraphs:
            for r in p.runs:
                r.font.color.rgb =
docx.shared.RGBColor.from_string(font_color)

    if font_size:
        for p in cell.paragraphs:
            for r in p.runs:
                r.font.size = docx.shared.Pt(font_size)

    if bold is not None:
        for p in cell.paragraphs:
            for r in p.runs:
                r.bold = bold

    if italic is not None:
        for p in cell.paragraphs:
            for r in p.runs:
                r.italic = italic

def set_repeat_table_header(row):
    """ set repeat table row on every new page
    """
    tr = row._tr
    trPr = tr.get_or_add_trPr()
    tblHeader = OxmlElement('w:tblHeader')
    tblHeader.set(qn('w:val'), "true")
    trPr.append(tblHeader)
    return row
```



```
def set_cell_margins(cell, **kwargs):
    """
    cell: actual cell instance you want to modify
    usage:
        set_cell_margins(cell, top=50, start=50, bottom=50, end=50)

    provided values are in twentieths of a point (1/1440 of an
    inch).
    read more here: http://officeopenxml.com/WPtableCellMargins.php
    """
    tc = cell._tc
    tcPr = tc.get_or_add_tcPr()
    tcMar = OxmlElement('w:tcMar')

    for m in ["top", "start", "bottom", "end"]:
        if m in kwargs:
            node = OxmlElement("w:{}".format(m))
            node.set(qn('w:w'), str(kwargs.get(m)))
            node.set(qn('w:type'), 'dxa')
            tcMar.append(node)

    tcPr.append(tcMar)

def add_table_to_doc_new(doc, df, heading, table_style='Table Grid',
                        txt="EDA1"):
    for p in doc.paragraphs:
        if txt in p.text:
            print("Gefunden: ", txt)
            ## Create the heading in here
            doc.add_heading(heading,
                            level=1).paragraph_format.alignment = WD_ALIGN_PARAGRAPH.CENTER
            ## Create the table in here
            #table = document.add_table(rows=len(data)+1,
            cols=len(headers))

            level=1).paragraph_format.alignment = WD_ALIGN_PARAGRAPH.CENTER
            columns = list(df.columns)
            print(columns)
            for col in columns:
                df[col].fillna(" ", inplace = True)

                # add table
                endcol=len(columns)
                table = doc.add_table(rows=1, cols=endcol,
                                      style=table_style)
                set_repeat_table_header(table.rows[0])
                table.autofit = True
                # add columns (if there is '_' then replace with space)
                for col in range(len(columns)):
                    set_cell_margins(table.cell(0, col), top=100,
                                     start=100, bottom=100, end=50)
                    table.cell(0, col).text = columns[col].replace("_",
                                     " ").capitalize()
                    # add data
                    for i, row in enumerate(df.itertuples()):
                        table_row = table.add_row().cells
                        for col in range(len(columns)):
                            #set_cell_margins(table_row[col], top=100,
                            start=100, bottom=100, end=50)
                            set_cell_margins(table_row[col], top=10,
                                             start=10, bottom=10, end=5)
                            table_row[col].text = str(row[col+1])
                            if i==1:
                                coll=table.cell(i, 1).text
```



```
nr=1
col2=table.cell(i, 2).text
if table.cell(i, col).text=="nan":
    table.cell(i, col).text==" "
#if i > 1 and col==table.cell(i-1, col).text:
#    table.cell(i, endcol).text=str(nr)

for row in range(df.shape[0]):
    for col in range(df.shape[-1]):
        #table.cell(row+1, col).text =
str(input_df.values[row, col])
        #table.cell(row, col).width = widths[col]
        #table.cell(row, col).height = Cm(1)
        table.cell(row+1,
col).paragraphs[0].paragraph_format.alignment =
WD_TABLE_ALIGNMENT.LEFT
        change_table_cell(table.rows[row+1].cells[col],
background_color="lightgreen", font_color="0000ff", font_size=8,
bold=True, italic=True)
        #change_table_cell(table.rows[row+1].cells[col],
background_color="White", font_color="000000", font_size=10,
bold=False, italic=True)
        table.rows[row].height_rule =
WD_ROW_HEIGHT_RULE.EXACTLY

    doc.add_section(WD_SECTION.ODD_PAGE)
    return doc

def move_table_after(table, paragraph):
    tbl = table._tbl
    paragraph.add_run().element.addnext(tbl)

# Copy Table from Excel
def add_table_to_doc(doc, df, heading, table_style='Table Grid'):
    doc.add_heading(heading, level=1).paragraph_format.alignment =
WD_ALIGN_PARAGRAPH.CENTER
    columns = list(df.columns)
    print(columns)
    for col in columns:
        df[col].fillna(" ", inplace = True)

    # add table
    endcol=len(columns)
    table = doc.add_table(rows=1, cols=endcol, style=table_style)
    #set_repeat_table_header(table.rows[0])
    table.autofit = True
    # add columns (if there is '_' then replace with space)
    for col in range(len(columns)):
        set_cell_margins(table.cell(0, col), top=100, start=100,
bottom=100, end=50)
        table.cell(0, col).text = columns[col].replace("_", " ")
    ".capitalize()"
    # add data
    for i, row in enumerate(df.itertuples()):
        table_row = table.add_row().cells
        for col in range(len(columns)):
            #set_cell_margins(table_row[col], top=100, start=100,
bottom=100, end=50)
            set_cell_margins(table_row[col], top=10, start=10,
bottom=10, end=5)
            table_row[col].text = str(row[col+1])
        if i==1:
            col1=table.cell(i, 1).text
            nr=1
            col2=table.cell(i, 2).text
```



```
if table.cell(i, col).text=="nan":  
    table.cell(i, col).text=" "  
#if i > 1 and col==table.cell(i-1, col).text:  
    #table.cell(i, endcol).text=str(nr)  
  
for row in range(df.shape[0]):  
    for col in range(df.shape[-1]):  
        #table.cell(row+1, col).text = str(input_df.values[row,  
col])  
        #table.cell(row, col).width = widths[col]  
        #table.cell(row, col).height = Cm(1)  
        table.cell(row+1,  
col).paragraphs[0].paragraph_format.alignment =  
WD_TABLE_ALIGNMENT.LEFT  
        #change_table_cell(table.rows[row+1].cells[col],  
background_color="lightgreen", font_color="0000ff", font_size=8,  
bold=True, italic=True)  
        change_table_cell(table.rows[row+1].cells[col],  
background_color="White", font_color="000000", font_size=10,  
bold=False, italic=True)  
        table.rows[row].height_rule = WD_ROW_HEIGHT_RULE.EXACTLY  
  
doc.add_section(WD_SECTION.ODD_PAGE)  
return doc  
  
def tab_copy(k):  
    #table = form.tables[k]  
    table = doc.tables[k]  
    table.alignment = WD_TABLE_ALIGNMENT.LEFT  
    tbl = table._tbl  
    paragraph = doc.add_paragraph("Hier wird die Tabelle №" +  
str(k+1) + " kopiert! Alle Daten wurde Simuliert!")  
    paragraph._p.addnext(tbl)  
  
def set_column_width(table, column, width_mm):  
    table.allow_autofit = False  
    for row in table.rows:  
        row.cells[column].width = Mm(width_mm)  
  
def set_cell_border(cell, **kwargs):  
    tc = cell._tc  
    tcPr = tc.get_or_add_tcPr()  
  
    # check for tag existnace, if none found, then create one  
    tcBorders = tcPr.first_child_found_in("w:tcBorders")  
    if tcBorders is None:  
        tcBorders = OxmlElement('w:tcBorders')  
        tcPr.append(tcBorders)  
  
    for edge in ('left', 'top', 'right', 'bottom', 'insideH',  
'insideV'):  
        edge_data = kwargs.get(edge)  
        if edge_data:  
            tag = 'w:{}'.format(edge)  
  
            # check for tag existnace, if none found, then create  
one  
            element = tcBorders.find(qn(tag))  
            if element is None:  
                element = OxmlElement(tag)  
                tcBorders.append(element)  
  
            # looks like order of attributes is important  
            for key in ["sz", "val", "color", "space", "shadow"]:  
                if key in edge_data:
```



```
        element.set(qn('w:{}').format(key)),
str(edge_data[key]))\n\n    with open(EDA_JSON, "w", encoding="utf-8") as file_handle:
        json.dump(EDA, file_handle, indent=4)\n\n#with open(EDA_JSON, 'r') as json_file:
##    json_object = json.load(json_file)
#print(json.dumps(json_object, indent=1))\n\nimport win32com.client
import time
start_time = time.time()
#print("InEDA_File: ", InEDA_File)
print("InEDA: ", InEDA)
word_app = win32com.client.Dispatch("Word.Application")
word_app.Visible = True
#doc = word_app.Documents.Open(InEDA_File)
doc = word_app.Documents.Open(InEDA)
print(len(doc.Paragraphs))
start = []
stop = []
to_start = "xStart"
to_stop = "xStop"\n\nfor i, p in enumerate(doc.Paragraphs):
    if to_start in p.Range.Text:
        #print(f"Text found in paragraph number #{i+1}")
        start.extend([i+1])
        #print("Start: ", start)
    if to_stop in p.Range.Text:
        #print(f"Text found in paragraph number #{i+1}")
        stop.extend([i+1])
        #print("Stop: ", stop)\n\nprint(len(start))\n\nfrom docx import Document
from docx.shared import Inches
import json\n\n# Используемые стили
HEADER_STYLE = "BoldHeader"
HEADER_LINK_STYLE = "BoldHeaderHyperlink"
CONTENT_STYLE = "Content"
CODE_STYLE = "Code"\n\ndoc = Document()
InDoc = Document(InEDA)
config = getConfigFile(EDA_JSON)\n\nsections = doc.sections
doc.add_heading('Исследовательский анализ данных (EDA)', 0)\n\ntasks = config["tasks"]
i=-1
k=-1
jj=-1
for task in tasks:
    i=i+1
    jj=jj+1
    #print("task: ", task)
    print("folder: ", task["folder"])
    if task['topic'] != " ":
        header = f"{task['topic']}"\n
```



```
Hstyle = f"{{task['Header']}}"
#print(Hstyle)
if header != " ":
    if Hstyle == "Heading2":
        doc.add_heading(header, 2)
    if Hstyle == "Heading3":
        doc.add_heading(header, 3)

if (task["Text"] == "y") and (i<len(start)):
#if (task["Text"] == "y"):
    b=start[i]
    e=stop[i]
    #print("folder: ", task["folder"])
    #print("Start: ", b, "Stop: ", e)
    ##print("i: ", i, "len(start): ", len(start))

    j=0
    for j in range(b, e-1):
        #print(InDoc.paragraphs[j])
        #get_para_data(doc, InEDA.paragraphs[j])
        get_para_data(doc, InDoc.paragraphs[j])

    if task[ "models"] == "gr":
        path=IMAGE+'\EDA' + str(i-2) + '.png'
        if os.path.isfile(path):
            print(path)
            doc.add_picture((IMAGE+'\EDA') + str(i-2) + '.png',
width=Inches(5.0))

    if task["Tab"] != " ":
        doc.add_paragraph("Таблица №" + str(jj))
#str(task["Tab"]))
        #doc.add_paragraph("EDA-Tab" + str(jj))
#str(task["Tab"]))
        #print("EDA-Tab" + str(jj))

    if task["sec"] == "y":
        doc.add_section(WD_SECTION.ODD_PAGE)

print(ReportEDA)
doc.save(ReportEDA)

from docx import Document
tabdoc="C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\PRODUKTION\OUTPUT\EDA-
Report.DOCX"

#document = Document(ReportEDAr)
document = Document(tabdoc)
doc = Document(ReportEDA)
print("ReportEDAr: ", ReportEDAr)
print("tabdoc: ", tabdoc)
count=len(document.tables)
print(count)
print("ReportEDA: ", ReportEDA)
print("ReportAll: ", ReportAll)

for num in range(0, count-1):
    template = document.tables[num]
    tbl = template._tbl
    new_tbl = deepcopy(tbl)
    #time.sleep(1)

    for para in doc.paragraphs:
        if para.text == "Таблица №" + str(num+1):
            print(para.text)
```



```
para._p.addnext(new_tbl)
time.sleep(1)

doc.save(ReportAll)
doc.save(ReportEDAr)

document = Document()
doc = Document(ReportEDAr)

for table in doc.tables:
    table.alignment = WD_TABLE_ALIGNMENT.CENTER
    table.autofit = False
    table.allow_autofit = False
    n_rows=len(table.rows)
    n_cols=len(table.columns)

    #table.cell(0, 0).text = 'Classes+Metrics'
    table.add_row()
    #set_column_width(table, 0, 35)
    #for c in range(1, 5):
    #    set_column_width(table, c, 20)

    g = table.cell(n_rows, 0)
    h = table.cell(n_rows, n_cols-1)
    g.merge(h)
    cell = table.cell(n_rows, n_cols-1)

    cell.paragraphs[0].paragraph_format.space_before = Inches(0)
    cell.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.LEFT
    cell.paragraphs[0].add_run("© Dr. Alexander Wagner. Все права
охраняются законом")
    change_table_cell(table.rows[n_rows].cells[2],
background_color="lightgreen", font_color="0000ff", font_size=8,
bold=True, italic=True)

    #table.style = 'Table Grid'
    for row in table.rows:
        row.height = Cm(0.55)
        row.height_rule = WD_ROW_HEIGHT_RULE.EXACTLY

    #table.rows[0].height = Cm(0.6)
    #table.rows[0].height_rule = WD_ROW_HEIGHT_RULE.EXACTLY

    table.rows[n_rows-1].height = Cm(0.45)
    table.rows[n_rows-1].height_rule = WD_ROW_HEIGHT_RULE.EXACTLY

doc.save(ReportEDAr)
print(ReportEDAr + str(" fertig!"))
```



## Аналитический Блок. Проведение ML Анализа и сохранение выходных результатов

```
import os, sys, inspect, time, datetime
warnings.filterwarnings('ignore')
stop=18
timestart = datetime.datetime.now()
date_time = timestart.strftime("%d.%m.%Y %H:%M:%S")

matrix ="data/heard.csv"
df = pd.read_csv(matrix)

df.rename({'Y': 'target'}, axis=1, inplace=True)
df = df.fillna(0)

train, test = train_test_split(df, test_size = 0.4)

X_train = train[train.columns.difference(['target'])]
y_train = train['target']

X_test = test[test.columns.difference(['target'])]
y_test = test['target']

cv_n_split = 2
random_state = 0
test_train_split_part = 0.2

train0, test0 = X_train, X_test
target0 = y_train
train, test, target, target_test = train_test_split(train0, target0,
test_size=test_train_split_part, random_state=random_state)

#st.write("3. Modellenauswahl")
cv_train = ShuffleSplit(n_splits=cv_n_split,
test_size=test_train_split_part, random_state=random_state)

metrics_all = {1 : 'r2_score', 2: 'acc', 3 : 'rmse', 4 : 're'}
metrics_now = [1, 2, 3, 4]

# list of accuracy of all model - amount of metrics_now * 2 (train &
test datasets)
num_models = 18
acc_train = []
acc_test = []
ShuffleSplit(n_splits=2, random_state=0, test_size=0.2,
train_size=None)

# GradientBoostingClassifier
mGBC=GradientBoostingClassifier()
GBC=GradientBoostingClassifier()

# KNeighborsClassifier
mKNC=KNeighborsClassifier()
KNC=KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p =
2)

# SVC
mSVC = SVC()
SVC = SVC(kernel="rbf", C=0.025, probability=True)

# DecisionTreeClassifier
mDTC=DecisionTreeClassifier()
DTC=DecisionTreeClassifier(criterion = 'entropy', random_state = 51)
```



```
# RandomForestClassifier
mRF=RandomForestClassifier()
RF=RandomForestClassifier(n_estimators = 20, criterion = 'entropy',
random_state = 51)

# XGBClassifier
mXGBC=XGBClassifier()
XGBC=XGBClassifier()

# AdaBoostClassifier
mAdaBoost=AdaBoostClassifier()
AdaBoost=AdaBoostClassifier(DecisionTreeClassifier(criterion =
'entropy', random_state = 200),
                           n_estimators=2000,
                           learning_rate=0.1,
                           algorithm='SAMME.R',
                           random_state=1,)

# LogisticRegression LR
LogReg=LogisticRegression()
LR=LogisticRegression(random_state = 51)

# Linear Regression
param_grid = {'C': np.linspace(.1, 1.5, 15)}
linreg = LogisticRegression()
linreg_CV = GridSearchCV(linreg, param_grid=param_grid, cv=cv_train,
verbose=False)

# Linear SVR
linear_svc = LinearSVC()
linear_svc_CV = GridSearchCV(linear_svc, param_grid={}, cv=cv_train,
verbose=False)

mlp = MLPClassifier()
param_grid = {'hidden_layer_sizes': [i for i in range(2,5)],
             'solver': ['sgd'],
             'learning_rate': ['adaptive'],
             'max_iter': [1000]
            }
mlp_GS = GridSearchCV(mlp, param_grid=param_grid, cv=cv_train,
verbose=False)

# Decision Tree Classifier
decision_tree = DecisionTreeClassifier()
decision_tree_CV = GridSearchCV(decision_tree, param_grid={},
cv=cv_train, verbose=False)

# Stochastic Gradient Descent
sgd = SGDClassifier()
sgd_CV = GridSearchCV(sgd, param_grid={}, cv=cv_train,
verbose=False)

# Ridge Classifier
ridge = RidgeClassifier()
ridge_CV = GridSearchCV(estimator=ridge, param_grid={'alpha':
np.linspace(.1, 1.5, 15)}, cv=cv_train, verbose=False)

# Bagging Classifier
bagging = BaggingClassifier()
bagging_CV = GridSearchCV(estimator=bagging, param_grid={},
cv=cv_train, verbose=False)

# AdaBoost Classifier
Ada_Boost = AdaBoostClassifier()
```



```
Ada_Boost_CV = GridSearchCV(estimator=Ada_Boost,
param_grid={'learning_rate' : [.01, .1, .5, 1]}, cv=cv_train,
verbose=False)

# LogisticRegression
logreg = LogisticRegression()
logreg_CV = GridSearchCV(estimator=logreg, param_grid={'C' : [.1,
.3, .5, .7, 1]}, cv=cv_train, verbose=False)

# Gaussian Naive Bayes
gaussian = GaussianNB()
gaussian_CV = GridSearchCV(estimator=gaussian, param_grid={}, cv=cv_train, verbose=False)

# Perceptron
perceptron = Perceptron()
perceptron_CV = GridSearchCV(estimator=perceptron, param_grid={}, cv=cv_train, verbose=False)

classifiers = [
    linreg_CV,
    logreg_CV,
    perceptron_CV,
    linear_svc_CV,
    mlp_GS,
    decision_tree_CV,
    sgd_CV,
    ridge_CV,
    bagging_CV,
    Ada_Boost_CV,
    GBC,
    KNC,
    DTC,
    RF,
    XGBC,
    AdaBoost,
    gaussian_CV,
    SVC
]

models = pd.DataFrame({'Model': [
    'Linear Regression',
    'Logistic Regression',
    'Perceptron',
    'Linear SVC',
    'MLPClassifier',
    'Decision Tree Classifier 1',
    'Stochastic Gradient Descent',
    'RidgeClassifier',
    'BaggingClassifier',
    'AdaBoostClassifier 1',
    'GradientBoostingClassifier',
    'KNeighborsClassifier',
    'DecisionTreeClassifier 2',
    'RandomForestClassifier',
    'XGBClassifier',
    'AdaBoostClassifier 2',
    'Naive Bayes',
    'SVC' ]})

i=-1
result_table = pd.DataFrame(columns=['classifiers',
                                      'fpr',
                                      'tpr',
                                      'accurate',
```



'auc'])

```
for classifier in classifiers:
    i=i+1
    name_model = models.iloc[i]['Model']

    if i < stop:
        pipe = Pipeline(steps=[('classifier', classifier)])
        model = pipe.fit(X_train, y_train)
        acc_metrics_calc(i,model,train,test,target,target_test)

        title = "Model: " + name_model
        figure, axes = plt.subplots(2, 1, figsize=(20, 10))

        if axes is None:
            _, axes = plt.subplots(1, 2, figsize=(20, 5))

        axes[0].set_title(title)
        axes[0].set_xlabel("Training examples")
        axes[0].set_ylabel("Score")

        cv_train = ShuffleSplit(n_splits=cv_n_split,
                                test_size=test_train_split_part, random_state=random_state)
        train_sizes, train_scores, test_scores, fit_times, _ = \
            learning_curve(estimator=model, X=train, y=target,
                           cv=cv_train,
                           train_sizes=np.linspace(.1, 1.0, 5),
                           return_times=True)
        train_scores_mean = np.mean(train_scores, axis=1)
        train_scores_std = np.std(train_scores, axis=1)
        test_scores_mean = np.mean(test_scores, axis=1)
        test_scores_std = np.std(test_scores, axis=1)
        fit_times_mean = np.mean(fit_times, axis=1)
        fit_times_std = np.std(fit_times, axis=1)

        axes[0].grid()
        axes[0].fill_between(train_sizes, train_scores_mean -
                            train_scores_std,
                            train_scores_mean + train_scores_std,
                            alpha=0.1,
                            color="r")
        axes[0].fill_between(train_sizes, test_scores_mean -
                            test_scores_std,
                            test_scores_mean + test_scores_std,
                            alpha=0.1,
                            color="g")
        axes[0].plot(train_sizes, train_scores_mean, 'o-',
                     color="r",
                     label="Training score")
        axes[0].plot(train_sizes, test_scores_mean, 'o-', color="g",
                     label="Cross-validation score")
        axes[0].legend(loc="best")

        # Plot n_samples vs fit_times
        axes[1].grid()
        axes[1].plot(train_sizes, fit_times_mean, 'o-')
        axes[1].fill_between(train_sizes, fit_times_mean -
                            fit_times_std,
                            fit_times_mean + fit_times_std,
                            alpha=0.1)
        axes[1].set_xlabel("Training examples")
        axes[1].set_ylabel("fit_times")
        axes[1].set_title("Scalability of the model")
```



```
    figure.savefig('assets/plot_learning_curve' + str(i) +
'.png')

    ypred = model.predict(X_test)

    if (i==2):
        clf = CalibratedClassifierCV(perceptron_CV)
        clf.fit(X_train, y_train)
        y_pred = clf.predict_proba(X_test) [:,:,1]
    elif (i==3):
        #svm = LinearSVC()
        clf = CalibratedClassifierCV(linear_svc_CV) #svm
        clf.fit(X_train, y_train)
        y_pred = clf.predict_proba(X_test) [:,:,1]
    elif (i==6):
        clf = CalibratedClassifierCV(sgd_CV)
        clf.fit(X_train, y_train)
        y_pred = clf.predict_proba(X_test) [:,:,1]
    elif (i==7):
        clf = CalibratedClassifierCV(ridge_CV)
        clf.fit(X_train, y_train)
        y_pred = clf.predict_proba(X_test) [:,:,1]
    else:
        y_pred = model.predict_proba(X_test) [:,:,1]

fpr, tpr, _ = roc_curve(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred).round(3)
accuracy = accuracy_score(ypred, y_test).round(2)
F1_Score = f1_score(y_test, ypred).round(2)

Precision_Score = precision_score(y_test, ypred).round(2)
Recall_Score = recall_score(y_test, ypred).round(2)
Balanced_Accuracy_Score = balanced_accuracy_score(y_test,
ypred).round(2)
target_names = ['class 0', 'class 1']
report = classification_report(y_test, ypred,
target_names=target_names, output_dict=True, digits=4)
print(report)
CR = pd.DataFrame(report).transpose()
print(CR)
CR.to_csv(f'data/CLSB_{i}.csv')

Accurate_train=pipe.score(X_train, y_train).round(2)
Accurate_test =pipe.score(X_test, y_test).round(2)
cm = confusion_matrix(y_test, ypred)
fig=plt.figure(figsize=(8,5))
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
sns.heatmap(cm, annot = True)
fig.savefig('assets/Heatmap' + str(i) + '.png')

Roc_Auc_Score = roc_auc_score(y_test, y_pred).round(2)

fig2=plt.figure(figsize=(8,5))
plt.style.use('seaborn-v0_8-darkgrid')
plt.plot(fpr, tpr, label='{} ROC (area = {:.2f})' %
(name_model, auc))
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1-Specificity(False Positive Rate)')
plt.ylabel('Sensitivity(True Positive Rate)')

plt.legend(loc="lower right")
```



```
plt.show()
fig2.savefig('assets/PlotROC' + str(i) + '.png')

i=-1
plt.style.use('seaborn-v0_8-darkgrid')
figC = plt.figure(figsize=(16,10))
#img = fig2img(figC)
#img.save('assets/PlotRoc.png')

for classifier in classifiers:
    i=i+1
    name_model = models.iloc[i]['Model']

    if i < stop:
        pipe = Pipeline(steps=[('classifier', classifier)])
        model = pipe.fit(X_train, y_train)
        ypred = model.predict(X_test)
        if (i==2):
            clf = CalibratedClassifierCV(perceptron_CV)
            clf.fit(X_train, y_train)
            y_pred = clf.predict_proba(X_test) [:,:,1]
        elif (i==3):
            #svm = LinearSVC()
            clf = CalibratedClassifierCV(linear_svc_CV) #svm
            clf.fit(X_train, y_train)
            y_pred = clf.predict_proba(X_test) [:,:,1]
        elif (i==6):
            clf = CalibratedClassifierCV(sgd_CV)
            clf.fit(X_train, y_train)
            y_pred = clf.predict_proba(X_test) [:,:,1]
        elif (i==7):
            clf = CalibratedClassifierCV(ridge_CV)
            clf.fit(X_train, y_train)
            y_pred = clf.predict_proba(X_test) [:,:,1]
        else:
            y_pred = model.predict_proba(X_test) [:,:,1]

        fpr, tpr, _ = roc_curve(y_test, y_pred)
        auc = roc_auc_score(y_test, y_pred).round(3)
        accuracy = accuracy_score(ypred, y_test).round(2)
        F1_Score = f1_score(y_test, ypred).round(2)

        Precision_Score = precision_score(y_test, ypred).round(2)
        Recall_Score = recall_score(y_test, ypred).round(2)
        Balanced_Accuracy_Score = balanced_accuracy_score(y_test,
        ypred).round(2)
        Classification_Report = classification_report(y_test, ypred)
        Accurate_train=pipe.score(X_train, y_train).round(2)
        Accurate_test =pipe.score(X_test, y_test).round(2)

        plt.plot(fpr, tpr, label='%s ROC (area = %0.2f)' %
        (name_model, auc))
        plt.plot([0, 1], [0, 1],'r--')
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('1-Specificity(False Positive Rate)')
        plt.ylabel('Sensitivity(True Positive Rate)')
        plt.title('Receiver Operating Characteristic')
        plt.legend(loc="lower right")

img = fig2img(figC)
img.save('assets/PlotRoc.png')

#6. Speicher der Ergebnisse
```



## Автоматизация Система Научных Исследований в медицине и здравоохранении «АСНИ-МЕД»

```
metricsnow = pd.DataFrame(metrics_now)
metricsnow.to_csv('data/Nostalgi2023_metrics_now.csv', index=False)

accall= pd.DataFrame(acc_all)
#allpred2023.to_csv('data/Nostalgi2023_acc_all.csv', index=False)
accall.to_csv('data/RT4.csv', index=False)

models = pd.DataFrame({'Model': [
    'Linear Regression',
    'Logistic Regression',
    'Perceptron',
    'Linear SVC',
    'MLPClassifier',
    'Decision Tree Classifier 1',
    'Stochastic Gradient Decent',
    'RidgeClassifier',
    'BaggingClassifier',
    'AdaBoostClassifier 1',
    'GradientBoostingClassifier',
    'KNeighborsClassifier',
    'DecisionTreeClassifier 2',
    'RandomForestClassifier',
    'XGBClassifier',
    'AdaBoostClassifier 2',
    'Naive Bayes',
    'SVC' ]})

metrics_all = {1 : 'r2_score', 2: 'acc', 3 : 'rmse', 4 : 're'}
metrics_now = [1, 2, 3, 4]

models=models[0:stop]
for x in metrics_now:
    xs = metrics_all[x]
    models[xs + '_Train'] = acc_all[(x-1)*2]
    models[xs + '_Test'] = acc_all[(x-1)*2+1]
    if xs == "acc":
        models[xs + '_Diff'] = models[xs + '_Train'] - models[xs + '_Test']

ms = metrics_all[metrics_now[1]] # the accuracy
models.sort_values(by=[(ms + '_Test'), (ms + '_Train')], ascending=False)
pd.options.display.float_format = '{:,.2f}'.format

# Plots
plt.style.use('seaborn-v0_8-darkgrid')

i=0
for x in metrics_now:
    i=i+1
    #fig = plt.figure(figsize=(4,10))
    xs = metrics_all[x]
    xs_train = metrics_all[x] + '_Train'
    xs_test = metrics_all[x] + '_Test'
    figD = plt.figure(figsize=[25,6])
    xx = models['Model']
    plt.tick_params(labelsize=14)
    plt.plot(xx, models[xs_train], label = xs_train)
    plt.plot(xx, models[xs_test], label = xs_test)
    plt.legend()
    plt.title(str(xs) + ' criterion for ' + str(num_models) + ' popular models for train and test datasets')
    plt.xlabel('Models')
    plt.ylabel(xs + ', %')
    plt.xticks(xx, rotation='vertical')
```



```
img = fig2img(figD)
img.save("assets/PlotE" + str(i) + ".png")

metrics_main = 2
xs = metrics_all[metrics_main]
xs_train = metrics_all[metrics_main] + '_Train'
xs_test = metrics_all[metrics_main] + '_Test'
direct_sort = False if (metrics_main >= 2) else True

models_sort = models.sort_values(by=[xs_test, xs_train],
                                 ascending=direct_sort)
models_best = models_sort[(models_sort.acc_Diff < 5) &
                           (models_sort.acc_Train > 90)]
models_best[['Model', ms + '_Train', ms + '_Test',
              'acc_Diff']].sort_values(by=['acc_Test'], ascending=False)
print("models_best RT1: eins!", models_best)
models_best.to_csv('data/RT1.csv', index=False)

metrics_all = {1 : 'r2_score', 2: 'acc', 3 : 'rmse', 4 : 're'}
metrics_now = [1, 2, 3, 4]

# list of accuracy of all model - amount of metrics_now * 2 (train &
# test datasets)
num_models = 18
acc_train = []
acc_test = []
acc_all = np.empty((len(metrics_now)*2, 0)).tolist()

models = pd.DataFrame({'Model': [
    'Linear Regression',
    'Logistic Regression',
    'Perceptron',
    'Linear SVC',
    'MLPClassifier',
    'Decision Tree Classifier 1',
    'Stochastic Gradient Decent',
    'RidgeClassifier',
    'BaggingClassifier',
    'AdaBoostClassifier 1',
    'GradientBoostingClassifier',
    'KNeighborsClassifier',
    'DecisionTreeClassifier 2',
    'RandomForestClassifier',
    'XGBClassifier',
    'AdaBoostClassifier 2',
    'SVC'
]}))

models_pred = pd.DataFrame(models.Model, columns = ['Model'])
N_best_models = len(models_pred.Model)

from sklearn.svm import SVC
i=0
for i in range(N_best_models):
    if i < N_best_models+1:
        name_model = models_pred.iloc[i]['Model']
        ii=i+1
        # model from Sklearn
        model = model_fit(name_model, train0, target0)
        acc_metrics_calc_pred(i, model, name_model, train0, test0,
                              target0)

models_pred=models_pred[0:N_best_models]

i=0
```



## Автоматизация Система Научных Исследований в медицине и здравоохранении «ASNI-MED»

```
for x in metrics_now:
    xs = metrics_all[x]
    #Auswahl N_best_models(17 im Moment) Spalten aus der
    acc_all_pred!
    acc_all_pred2=acc_all_pred[x-1][0:N_best_models]
    models_pred[xs + '_train'] = acc_all_pred2

sort_pred=models_pred[['Model',
'acc_train']].sort_values(by=['acc_train'], ascending=False)
sort_pred.to_csv('data/RT2.csv', index=False)

models_pred=models_pred[0:N_best_models]
i=0
for x in metrics_now:
    xs = metrics_all[x]
    #Auswahl N_best_models(17 im Moment) Spalten aus der
    acc_all_pred!
    acc_all_pred2=acc_all_pred[x-1][0:N_best_models]
    models_pred[xs + '_train'] = acc_all_pred2

print(models_pred)
sort_pred=models_pred[['Model',
'r2_score_train','acc_train','rmse_train',
're_train']].sort_values(by=['acc_train'], ascending=False)
#sort_pred.to_csv('data/Nostalgi2023_BestModel.csv', index=False)
sort_pred.to_csv('data/RT3.csv', index=False)

import pandas as pd
#RT5=pd.read_csv("C:\IPYNBgesamt\ASNI-FEN\ASNI-FEN-
SYSTEM\data\pred2023.csv")

RT1=pd.read_csv("C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\data\RT1.csv")
RT2=pd.read_csv("C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\data\RT2.csv")
RT3=pd.read_csv("C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\data\RT3.csv")
RT4=pd.read_csv("C:\IPYNBgesamt\ASNI-FEN\ASNI-Report\data\RT4.csv")
RT5=pd.read_csv("C:\IPYNBgesamt\ASNI-FEN\ASNI-
Report\data\pred2023.csv")
RT5.to_csv('data/RT5.csv', index=False)

print("Programm Start: ", date_time)
timeend = datetime.datetime.now()
date_time = timeend.strftime("%d.%m.%Y %H:%M:%S")
print("Programm Finish:",date_time)
timedelta = round((timeend-timestamp).total_seconds(), 2)

r=(timeend-timestamp)
t=int(timedelta/60)
if timedelta-t*60 < 10:
    t2=":0" + str(int(timedelta-t*60))
else:
    t2=":" + str(int(timedelta-t*60))
txt="Общее время работы программы составляет: 00:" + str(t) + t2
print(txt)
```



## Заключительный системный Блок. Создание полного научного отчета

```
#System Generell
today = datetime.date.today()
year = today.year
cwd = os.getcwd()
template=os.path.join(cwd, "Templates")
CONFIG_JSON = os.path.join(cwd, "Templates\ASNIR.json")
ASSETS=os.path.join(cwd, "ASSETS")

Report_file = os.path.join(cwd, "ASNI_ReportV05R4.docx")
print("Report_file: ", Report_file)
print("CONFIG_JSON: ", CONFIG_JSON)
print("ASSETS: ", ASSETS)

now=datetime.datetime.now()
timestart = now.replace(microsecond=0)

#Def Section
def getConfigFile(config):
    with open(config, encoding='utf-8') as json_file:
        return json.load(json_file)

def addTextParagraphToDocumentInStyle(text, document, style):
    p = document.add_paragraph(text)
    p.style = document.styles[style]

def AddBookmark(t, mark):
    # Find a specific text or phrase in the document
    paragraph = section.AddParagraph()
    paragraph.Format.HorizontalAlignment =
HorizontalAlignment.Center
    text = paragraph.AppendText(t)
    text = document.FindString(t, False, True)
    # Get the found text as a single text range
    textRange = text.GetAsOneRange()
    # Get the paragraph where the text range is located
    paragraph = textRange.OwnerParagraph

    # Get the index position of the text in the paragraph
    index = paragraph.ChildObjects.IndexOf(textRange)

    # Add a bookmark start mark to the paragraph
    start = paragraph.AppendBookmarkStart(mark)
    # Insert the bookmark start mark at the index position of the
text range
    paragraph.ChildObjects.Insert(index, start)
    # Add a bookmark end mark to the paragraph
    end = paragraph.AppendBookmarkEnd(mark)
    # Insert the bookmark end mark after the text range
    paragraph.ChildObjects.Insert(index + 2, end)

def update_toc(docx_file):
    word = win32com.client.DispatchEx("Word.Application")
    word.Visible = 1
    word.DisplayAlerts = 0

    doc = word.Documents.Open(docx_file)
    #wd_section = doc.Sections(1)
    toc_count = doc.TablesOfContents.Count
    print(toc_count)
    stringG='INHALTSVERZEICHNIS'
    stringK='Содержание'
    if toc_count == 0:
```



```
for i, p in enumerate(doc.Paragraphs):
    if stringK in p.Range.Text:
        try:
            p.Range.InsertParagraphAfter()
            parag_range = doc.Paragraphs(i+2).Range
            parag_range.Font.Name = 'Arial'
            parag_range.Font.Size = 14
            parag_range.Font.Bold = constants.wdToggle
            parag_range.Font.Size = 12
            doc.TablesOfContents.Add(Range=parag_range,
                                    UseHeadingStyles=True,
                                    UpperHeadingLevel=1,
                                    LowerHeadingLevel=4)
        except Exception as e:
            print("Ja :", e, "Nein")
            break

    elif toc_count == 1:
        toc = doc.TablesOfContents(1)
        toc.Update()
        print('TOC should have been updated.')
    else:
        print('TOC has not been updated for sure...')

doc.Close()
word.Quit()

def update_tocHalb(doc):
    toc_count = doc.TablesOfContents.Count
    print(toc_count)
    stringG='INHALTSVERZEICHNIS'
    stringK='Содержание'
    if toc_count == 0:
        for i, p in enumerate(doc.Paragraphs):
            if stringK in p.Range.Text:
                try:
                    p.Range.InsertParagraphAfter()
                    parag_range = doc.Paragraphs(i+2).Range
                    parag_range.Font.Name = 'Arial'
                    parag_range.Font.Size = 14
                    parag_range.Font.Bold = constants.wdToggle
                    parag_range.Font.Size = 12
                    doc.TablesOfContents.Add(Range=parag_range,
                                            UseHeadingStyles=True,
                                            UpperHeadingLevel=1,
                                            LowerHeadingLevel=4)
                except Exception as e:
                    print("Ja :", e, "Nein")
                    break

    elif toc_count == 1:
        toc = doc.TablesOfContents(1)
        toc.Update()
        print('TOC should have been updated.')
    else:
        print('TOC has not been updated for sure...')

def set_column_width(table, column, width_mm):
    table.allow_autofit = False
    for row in table.rows:
        row.cells[column].width = Mm(width_mm)

def set_repeat_table_header(row):
    tr = row._tr
    trPr = tr.get_or_add_trPr()
```



```
tblHeader = OxmlElement('w:tblHeader')
tblHeader.set(qn('w:val'), "true")
trPr.append(tblHeader)
return row

def change_table_cell(cell, background_color=None, font_color=None,
font_size=None, bold=None, italic=None):
    if background_color:
        shading_elm = parse_xml(r'<w:shd {}'
w:fill="{}"/>'.format(nsdecls('w')), background_color))
cell._tc.get_or_add_tcPr().append(shading_elm)

    if font_color:
        for p in cell.paragraphs:
            for r in p.runs:
                r.font.color.rgb =
docx.shared.RGBColor.from_string(font_color)

    if font_size:
        for p in cell.paragraphs:
            for r in p.runs:
                r.font.size = docx.shared.Pt(font_size)

    if bold is not None:
        for p in cell.paragraphs:
            for r in p.runs:
                r.bold = bold

    if italic is not None:
        for p in cell.paragraphs:
            for r in p.runs:
                r.italic = italic

def set_cell_border(cell, **kwargs):
    tc = cell._tc
    tcPr = tc.get_or_add_tcPr()

    # check for tag existnace, if none found, then create one
    tcBorders = tcPr.first_child_found_in("w:tcBorders")
    if tcBorders is None:
        tcBorders = OxmlElement('w:tcBorders')
        tcPr.append(tcBorders)

    for edge in ('left', 'top', 'right', 'bottom', 'insideH',
'insideV'):
        edge_data = kwargs.get(edge)
        if edge_data:
            tag = 'w:{}{}'.format(edge)

            # check for tag existnace, if none found, then create
one
            element = tcBorders.find(qn(tag))
            if element is None:
                element = OxmlElement(tag)
                tcBorders.append(element)

            # looks like order of attributes is important
            for key in ["sz", "val", "color", "space", "shadow"]:
                if key in edge_data:
                    element.set(qn('w:{}{}'.format(key)),
str(edge_data[key]))


def delete_paragraph(paragraph):
    p = paragraph._element
```



```
p.getparent().remove(p)
p._p = p._element = None

def replace_copy(txt):
    finder = wordapp.Selection.Find
    finder.Text = txt
    finder.Execute()
    #wordapp.Selection.MoveLeft()
    #wordapp.Selection.MoveDown()
    wordapp.Selection.MoveStart
    wordapp.Selection.Paste()
    if txt=="KapEDA":
        time.sleep(10.0)

def Text_copy(file):
    wordapp =
win32com.client.gencache.EnsureDispatch("Word.Application")
    wordapp.Visible = True
    worddoc = wordapp.Documents.Open(file)
    worddoc.Select()
    wordapp.Selection.Copy()
    worddoc.ActiveWindow.Close()
    #wordapp.Application.Quit(-1)

def set_repeat_table_header(row):
    """ set repeat table row on every new page
        t = doc.add_table(n_rows+2, n_cols, style="Table Grid")
        set_repeat_table_header(t.rows[0])
    """
    tr = row._tr
    trPr = tr.get_or_add_trPr()
    tblHeader = OxmlElement('w:tblHeader')
    tblHeader.set(qn('w:val'), "true")
    trPr.append(tblHeader)
    return row

from docxtpl import DocxTemplate
doc = DocxTemplate("rTemplateTest.docx")
reportWordPath = 'ASNI_ReportTest01.docx'

Inmodels = pd.DataFrame({'Model': [
    'Linear Regression',
    'Logistic Regression',
    'Perceptron',
    'Linear SVC',
    'MLPClassifier',
    'Decision Tree Classifier 1',
    'Stochastic Gradient Decent',
    'RidgeClassifier',
    'BaggingClassifier',
    'AdaBoostClassifier 1',
    'GradientBoostingClassifier',
    'KNeighborsClassifier',
    'DecisionTreeClassifier 2',
    'RandomForestClassifier',
    'XGBClassifier',
    'AdaBoostClassifier 2',
    'Naive Bayes',
    'SVC' ]})

Nk=len(Inmodels)
print(Nk)

with open("Templates/ASNIR.json", "w", encoding="utf-8") as file_handle:
```



```
json.dump(Asni, file_handle, indent=4)

#Spire
from spire.doc.common import *
from spire.doc import *

CONFIG_JSON = "Templates/ASNIR.json"
# Используемые стили
HEADER_STYLE = "BoldHeader"
HEADER_LINK_STYLE = "BoldHeaderHyperlink"
CONTENT_STYLE = "Content"
CODE_STYLE = "Code"

document = Document()
section = document.AddSection()
paragraph = section.AddParagraph()
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
text = paragraph.AppendText("{Projekt}")
text.CharacterFormat.FontName = "Times New Roman"
text.CharacterFormat.FontSize = 26
text.CharacterFormat.Bold = True
text.CharacterFormat.TextColor = Color.get_Blue()

paragraph = section.AddParagraph()
paragraph = section.AddParagraph()
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
text = paragraph.AppendText("Тема исследования: {{Projekt3}}")
text.CharacterFormat.FontName = "Times New Roman"
text.CharacterFormat.FontSize = 16
text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()
paragraph = section.AddParagraph()
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
text = paragraph.AppendText("Проект: {{Thema}}")
text.CharacterFormat.FontName = "Times New Roman"
text.CharacterFormat.FontSize = 16
text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()
paragraph = section.AddParagraph()
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
text = paragraph.AppendText("Автор исследования: {{Forscher}}")
text.CharacterFormat.FontName = "Times New Roman"
text.CharacterFormat.FontSize = 16
text.CharacterFormat.Bold = True
for num in range(9):
    i=num+1
    paragraph = section.AddParagraph()

paragraph = section.AddParagraph()
paragraph = section.AddParagraph()
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
text = paragraph.AppendText("{logo}")
text.CharacterFormat.FontName = "Times New Roman"
text.CharacterFormat.FontSize = 16
text.CharacterFormat.Bold = True

for num in range(13):
    i=num+1
    paragraph = section.AddParagraph()

paragraph = section.AddParagraph()
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
```



```
text = paragraph.AppendText("{{Site}}")
text.CharacterFormat.FontName = "Times New Roman"
text.CharacterFormat.FontSize = 14
text.CharacterFormat.Bold = True

paragraph = section.AddParagraph()
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
text = paragraph.AppendText("{{Year}}")
text.CharacterFormat.FontName = "Times New Roman"
text.CharacterFormat.FontSize = 14
text.CharacterFormat.Bold = True

section = document.AddSection()
paragraph = section.AddParagraph()
paragraph = section.AddParagraph()

paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center
text = paragraph.AppendText("Содержание")
text.CharacterFormat.FontName = "Arial"
text.CharacterFormat.FontSize = 14
text.CharacterFormat.Bold = True
text.CharacterFormat.Italic = True

paragraph = section.AddParagraph()
text = paragraph.AppendText(" ")
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

#paragraph = section.AddParagraph()
paragraph.Format.HorizontalAlignment = HorizontalAlignment.Center

outputFile = "TemplateTOC.docx"
document.SaveToFile(outputFile, FileFormat.Docx)
document.Close()

from docx import Document, enum
doc = Document("TemplateTOC.docx")
lines = doc.paragraphs
for line in lines:
    #print(line)
    if "{{ClassBlk" in line.text or "{{Kap" in line.text:
        print(line.text)
        line.paragraph_format.first_line_indent = Inches(0.25)
        continue

doc.save("rReportTest.docx")

#im = ASSETS + '\ImageAll\GRAPH0.png'
#print(im)

doc = DocxTemplate("rReportTest.docx")
reportWordPath = 'ASNI_ReportTest01.docx'
Nk=18

with open("Templates/ASNIR.json", "w", encoding="utf-8") as file_handle:
    json.dump(Asni, file_handle, indent=4)

with open('Templates/ASNIR.json', 'r', encoding='utf-8') as file_object:
    ASNIDict = json.load(file_object)

ASNIDict['logo'] = InlineImage(doc, ASSETS +
    '\ImageAll\GRAPH0.png', Cm(12))

for num in range(Nk):
```



```
i=num+1
#print(i)
ASNI_dict['Heatmap'+str(num)] = InlineImage(doc, ASSETS +
'\ImageAll\Heatmap' + str(num) + '.png', Cm(18))
ASNI_dict['PltR' + str(num)] = InlineImage(doc, ASSETS +
'\ImageAll\PlotROC' + str(num) + '.png', Cm(18))
ASNI_dict['PltL' + str(num)] = InlineImage(doc, ASSETS +
'\ImageAll\plot_learning_curve' + str(num+1) + '.png', Cm(20))

#print(ASNI_dict)

doc.render(ASNI_dict)
doc.save(reportWordPath)

reportWordPath="ASNI_ReportTest01.docx"
doc.save("ASNI_ReportTest01.docx")

ReportTest01= os.path.join(cwd, "ASNI_ReportTest01.docx")
print("ReportTest01: ", ReportTest01)

word = win32com.client.gencache.EnsureDispatch("Word.Application")
word.Visible = True
#doc = word.Documents.Open("C:\IPYNBgesamt\ASNI-FEN\ASNI-
Report\ASNI_ReportTest01.docx")
doc = word.Documents.Open(ReportTest01)
i=1

for num in range(Nk):
    i=i+1
    df=pd.read_csv(f'data/CLSB_{num}.csv')
    rng = doc.Bookmarks("Table" + str(num)).Range

Table=rng.Tables.Add(rng, NumRows=df.shape[0]+1, NumColumns=df.shape[1])
for col in range(df.shape[1]):
    Table.Cell(1,col+1).Range.Text=str(df.columns[col])
    for row in range(df.shape[0]):
        Table.Cell(row+1+1,col+1).Range.Text=str(df.iloc[row,col])

doc.Close()
word.Quit()

print("Table in Order!")

document = Document()
doc = Document("ASNI_ReportTest01.docx")

for table in doc.tables:
    table.alignment = WD_TABLE_ALIGNMENT.CENTER
    table.autofit = False
    table.allow_autofit = False
    n_rows=len(table.rows)
    n_cols=len(table.columns)

    table.cell(0, 0).text = 'Classes+Metrics'
    table.add_row()
    set_column_width(table, 0, 35)
    for c in range(1, 5):
        set_column_width(table, c, 20)

    g = table.cell(n_rows, 0)
    h = table.cell(n_rows, n_cols-1)
```



```
g.merge(h)
cell = table.cell(n_rows, n_cols-1)

cell.paragraphs[0].paragraph_format.space_before = Inches(0)
cell.paragraphs[0].alignment = WD_PARAGRAPH_ALIGNMENT.LEFT
cell.paragraphs[0].add_run("© Dr. Alexander Wagner. Все права
охраняются законом")
change_table_cell(table.rows[n_rows].cells[2],
background_color="lightgreen", font_color="0000ff", font_size=8,
bold=True, italic=True)
table.style = 'Table Grid'

set_repeat_table_header(table.rows[0])

for i in range(1, n_rows):
    for j in range(1, n_cols):
        element=table.cell(i, j).text
        partition = element.partition('.')
        if (partition[0].isdigit() and partition[1] == '.' and
partition[2].isdigit()):
            newelement = float(element)
            y=round(newelement,3)
            table.cell(i, j).text=str(y)
            table.cell(i,
j).paragraphs[0].paragraph_format.alignment =
WD_TABLE_ALIGNMENT.RIGHT

        for c in range(0, n_cols):
            change_table_cell(table.rows[0].cells[c],
background_color="lightgreen", font_color="0000ff", font_size=12,
bold=True, italic=True)

                for cell in table.columns[c].cells:
                    cell.paragraphs[0].paragraph_format.space_after =
Inches(0)
                    cell.paragraphs[0].paragraph_format.space_before =
Inches(0)
                    cell.vertical_alignment =
WD_CELL_VERTICAL_ALIGNMENT.CENTER

                    set_cell_border(
                        cell,
                        top={"sz": 0.5, "val": "double", "color": "#000000",
"space": "0"},

                        bottom={"sz": 0.5, "val": "double", "color":
"#000000", "space": "0"},

                        left={"sz": 0.5, "val": "double", "color":
"#000000", "space": "0"},

                        right={"sz": 0.5, "val": "double", "color":
"#000000", "space": "0"},

                        insideH={"sz": 0.5, "val": "double", "color":
"#000000", "space": "0"},

                        end={"sz": 0.5, "val": "double", "color": "#000000",
"space": "0"})
                )

for row in table.rows:
    row.height = Cm(0.55)
    row.height_rule = WD_ROW_HEIGHT_RULE.EXACTLY

table.rows[0].height = Cm(0.6)
table.rows[0].height_rule = WD_ROW_HEIGHT_RULE.EXACTLY
```



## Автоматизированная Система Научных Исследований в медицине и здравоохранении «АСНИ-МЕД»

```
table.rows[n_rows-1].height = Cm(0.45)
table.rows[n_rows-1].height_rule = WD_ROW_HEIGHT_RULE.EXACTLY

doc.save("ASNI_Report.docx")
print("ASNI_Report.docx fertig!")
from spire.doc import *
document = Document()
file = os.path.join(cwd, "ASNI_Report.docx")
dt=datetime.datetime.fromtimestamp(os.stat(file).st_mtime)

txd = "Документ актуализирован: " + dt.strftime('%d.%m.%Y %H:%M:%S')
print("Mody:", txd)

# Load a Word document
document.LoadFromFile(file)
# Get the first section
section = document.Sections[0]

# Get header
header = section.HeadersFooters.Header

# Add a paragraph to the header and set its alignment style
headerParagraph = header.AddParagraph()
headerParagraph.Format.HorizontalAlignment =
HorizontalAlignment.Left
#headerParagraph.Format.VerticalAlignment = VerticalAlignment.Center
section.header_distance = Cm(1.2)

headerPicture =
headerParagraph.AppendPicture("ASSETS\ImageAll\logo2.jpg")
headerPicture.TextWrappingStyle = TextWrappingStyle.Square
headerPicture.VerticalOrigin = VerticalOrigin.Line
headerPicture.VerticalAlignment = ShapeVerticalAlignment.Center
#headerPicture.HorizontalAlignment = ShapeHorizontalAlignment.Right
headerPicture.HorizontalAlignment = ShapeHorizontalAlignment.Left
headerPicture.VerticalOrigin = VerticalOrigin.TopMarginArea

text = headerParagraph.AppendText("Автоматизированная Система Научных
Исследований в медицине и здравоохранении «АСНИ-МЕД»)
text.CharacterFormat.FontName = "Times New"
text.CharacterFormat.FontSize = 9
text.CharacterFormat.Bold = True
text.CharacterFormat.TextColor = Color.get_Blue()

section = document.Sections[0]
# Get footer
footer = section.HeadersFooters.Footer

# Add a paragraph to the footer paragraph and set its alignment
style
footerParagraph = footer.AddParagraph()
footerParagraph.Format.HorizontalAlignment =
HorizontalAlignment.Left
# Add text to the footer paragraph and set its font style
text = footerParagraph.AppendText("© Dr. Alexander Wagner, Все права
охраняются законом. " + txd)
text.CharacterFormat.FontName = "Times New"
text.CharacterFormat.FontSize = 9
text.CharacterFormat.Bold = True
text.CharacterFormat.TextColor = Color.get_Blue()

footerParagraph = footer.AddParagraph()
footerParagraph.Format.HorizontalAlignment =
HorizontalAlignment.Right
text = footerParagraph.AppendText("Page ")
```



## Автоматизация Система Научных Исследований в медицине и здравоохранении «АСНИ-МЕД»

```
txt1=footerParagraph.AppendField("page number", FieldType.FieldPage)
txt2=footerParagraph.AppendText(" of ")
txt3=footerParagraph.AppendField("number of pages",
FieldType.FieldNumPages)
text.CharacterFormat.TextColor = Color.get_Blue()
txt1.CharacterFormat.TextColor = Color.get_Blue()
txt2.CharacterFormat.TextColor = Color.get_Blue()
txt3.CharacterFormat.TextColor = Color.get_Blue()

# Save the result file
document.SaveToFile("AddFootnoteForParagraph.docx",
FileFormat.Docx2016)
document.Close()

from docx import Document
#import time
doc = Document("AddFootnoteForParagraph.docx")
s=len(doc.sections)
for nt in range(s):
    section = doc.sections[nt]
    header = doc.sections[nt].header
    footer = doc.sections[nt].footer

    section.header_distance = Cm(1.0)
    section.footer_distance = Cm(1.0)

    header_para = header.paragraphs[0]
    header_para.paragraph_format.space_before = Pt(0)
    header_para.paragraph_format.space_after = Pt(7)

    footer_para = footer.paragraphs[0]
    footer_para.paragraph_format.space_before = Pt(0)
    footer_para.paragraph_format.space_after = Pt(0)

    footer_para = footer.paragraphs[1]
    footer_para.paragraph_format.space_before = Pt(0)
    footer_para.paragraph_format.space_after = Pt(0)

section = doc.sections[0]
section.different_first_page_header_footer = True

lines = doc.paragraphs
n=-1
for line in lines:
    n=n+1
    if line.text == "Evaluation Warning: The document was created
with Spire.Doc for Python.":
        delete_paragraph(line)
        continue

reportWordPath="ASNI_ReportPre.docx"
doc.save("ASNI_ReportPre.docx")
print("ASNI_ReportPre.docx fertig!")

ReportPre = os.path.join(cwd, "ASNI_ReportPre.docx")
print("ReportPre: ", ReportPre)

KapE = os.path.join(cwd, "data\Kap")
print("KapE: ", KapE)

KapM = os.path.join(cwd, "data\Modr")
print("KapM: ", KapM)

ResultDoc = os.path.join(cwd, "ASNI_ReportResultV06.docx")
print("ResultDoc: ", ResultDoc)
```



```
KapM = os.path.join(cwd, "data\Modr")
print("KapM: ", KapM)

wordapp =
win32com.client.gencache.EnsureDispatch("Word.Application")
wordapp.Visible = True
wordapp.DisplayAlerts = 0
newdoc = wordapp.Documents.Open(ReportPre) #C:\IPYNBgesamt\ASNI-
FEN\ASNI-Report\ASNI_ReportPre.docx)

#for k in range(0, 13):
for k in range(0, 14):
    Text_copy(KapE + str(k) + ".docx") #r"C:\IPYNBgesamt\ASNI-
FEN\ASNI-Report\data\Kap" + str(k) + ".docx")
    time.sleep(4.4)
    KapTxt="Kap" + str(k)
    print(KapTxt)
    replace_copy(KapTxt)

newdoc.Select()
finder = wordapp.Selection.Find
finder.Text = "Предисловие"
finder.Execute()
wordapp.Selection.MoveLeft()

for num in range(Nk):
    print(KapM + str(num+1) + ".docx")
    Text_copy(KapM + str(num+1) + ".docx")
    time.sleep(4.4)
    MoTxt="ClassBlk" + str(num)
    print(num, " ", MoTxt)
    replace_copy(MoTxt)

time.sleep(4.4)
update_tocHalb(newdoc)

Report\ASNI_Report.docx")
newdoc.SaveAs(ResultDoc)
wordapp.Application.Quit(-1)

print("Programm Start: ", timestart)
txt="Programm Dauer: 00:" + str(t) + t2
print(txt)
----- End of File! -----
```