Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»

УТВЕРЖДАЮ
Директор Института магистратуры
_____Иванова Е.А.
«_23___» __июня_ 2023_г.

Рабочая программа дисциплины
Прикладное машинное обучение

Направление 01.04.02 Прикладная математика и информатика
магистерская программа 01.04.02.04 "Искусственный интеллект: математические модели и прикладные решения"

Для набора 2023 года

Квалификация
Магистр

Составитель программы:

Алексейчик Тамара Васильевна, к.э.н., доц, кафедры высшей фундаментальной и прикладной математики

I. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цели освоения дисциплины: изучение современных прикладных методов и технологий машинного обучения

Задачи:

- изучение основных понятиях задач машинного обучения: обучающая выборка, модель алгоритмов, метод обучения, функционал качества, проблема переобучения и обобщающая способность алгоритмов, вероятностная постановка задачи обучения;
- знакомство с методами решения задач классификации и регрессии: байесовским подходом (наивным, смесями многомерных нормальных распределений), метрическими алгоритмами, линейными алгоритмами (однослойным персептроном, логистической регрессией, SVM), логическими алгоритмами классификации, методами восстановления регрессии на основе SVD-разложения
- освоение методов оценки качества работы алгоритмов;
- знакомство с методами сбора и разметки данных
- изучение способов оценки информативности закономерностей и поиска информативных закономерностей
- приобретение навыков построения композиций алгоритмов
- получение представлений о современном состоянии машинного обучения
- знакомство с рекомендательными системами

II. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОП ВО

2.1. Учебная дисциплина «Прикладное машинное обучение» (1 год, 2 семестр) относится к части блока дисциплин (модулей), формируемых участниками образовательных отношений и является дисциплиной по выбору.

2.2. Для изучения данной учебной дисциплины необходимы знания, умения и навыки, формируемые предшествующими дисциплинами бакалавриата «Основы программирования», «Языки программирования», «Теория вероятностей и математическая статистика», «Алгебра и геометрия», «Вычислительная математика».

2.3. Знания и навыки, полученные в ходе изучения данной дисциплины, могут использоваться для решения профессиональных задач в научно-исследовательской, научно-производственной и проектной деятельности, в частности, при разработке алгоритмов и программ в рамках подготовки выпускной квалификационной работы.

III. ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Процесс изучения дисциплины направлен на формирование элементов следующих компетенций в соответствии с ФГОС ВО и ОП ВО по данному направлению подготовки (специальности):

Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы:

Шифр и формулировка компетенций (результаты освоения ОП)	**Шифр и формулировка индикаторов компетенций (результатов освоения ОП)**	**Элементы компетенций, формируемые дисциплиной**
ПК-3. Способен разрабатывать и применять методы и алгоритмы машинного обучения для решения задач	ПК-3.1. Ставит задачи по разработке или совершенствованию методов и алгоритмов для решения комплекса задач предметной области	ПК-3.1. З-1. Знает классы методов и алгоритмов машинного обучения ПК-3.1. У-1. Умеет ставить задачи и разрабатывать новые методы и алгоритмы машинного обучения ПК-3.1. Н-1. Владеет навыком оперирования основными понятиями машинного обучения: прецедент, обучающая выборка, признаки объектов, виды признаков, матрица объектов-признаков, модель алгоритмов, метод обучения, функционал качества алгоритма, проблема переобучения и обобщающая способность алгоритма. Навыком применения метрических алгоритмы классификации, использования EM-алгоритма разделения смеси, работы с алгоритмом стохастического градиента для минимизации функционала среднего риска, построения моделей по методу опорных векторов (SVM), работы с алгоритмами восстановления регрессии: метод наименьших квадратов, непараметрическая регрессия, многомерная линейная регрессия, подход с использованием SVD-разложения матрицы, гребневая регрессия, использования логических методов классификации, построения композиций алгоритмов
	ПК-3.2. Руководит исследовательской группой по разработке или совершенствованию методов и алгоритмов для решения комплекса задач предметной области	ПК-3.2. З-1. Знает методы и критерии оценки качества моделей машинного обучения ПК-3.2. У-1. Умеет определять критерии и методы оценки результатов моделирования при построении систем искусственного интеллекта в исследуемой области ПК-3.2. Н-1. Владеет эмпирическими оценками обобщающей способности алгоритма, алгоритмом построения кривой ошибок и вычисления AUC
	ПК-3.3. Разрабатывает унифицированные и обновляемые методологии описания, сбора и раз-	ПК-3.3. З-1. Знает унифицированные и обновляемые методологии описания, сбора и разметки данных, а также механизмы контроля за соблюдением указанных методологий ПК-3.3. У-1. Умеет разрабатывать унифицированные и

	метки данных, а также механизмы контроля за соблюдением указанных методологий	обновляемые методологии описания, сбора и разметки данных, а также механизмы контроля за соблюдением указанных методологий ПК-3.3. Н-1. Владеет навыком использования методов сбора и разметки данных
ПК-4. Способен руководить проектами по созданию комплексных систем искусственного интеллекта	ПК-4.1. Руководит разработкой архитектуры комплексных систем искусственного интеллекта	ПК-4.1. З-1. Знает возможности современных инструментальных средств и систем программирования для решения задач машинного обучения ПК-4.1. У-1. Умеет проводить сравнительный анализ и осуществлять выбор современных инструментальных средств для решения задач машинного обучения ПК-4.1. Н-1. Владеет навыком выбора и применения алгоритмов машинного обучения к конкретным задачам, навыком поиска оптимального алгоритма для вероятностной постановки задачи с учетом заданной метрики качества, методами построения рекомендательных систем, навыком применения языка программирования Python и библиотек машинного обучения
	ПК-4.2. Осуществляет руководство созданием комплексных систем искусственного интеллекта с применением новых методов и алгоритмов машинного обучения	ПК-4.2. З-1. Знает функциональность современных инструментальных средств и систем программирования в области создания моделей и методов машинного обучения ПК-4.2. З-2. Знает принципы построения систем искусственного интеллекта, методы и подходы к планированию и реализации проектов по создания систем искусственного интеллекта ПК-4.2. У-1. Умеет применять современные инструментальные средства и системы программирования для разработки новых методов и моделей машинного обучения ПК-4.2. У-2. Умеет руководить выполнение коллективной проектной деятельности для создания, поддержки и использования систем искусственного интеллекта ПК-4.2. Н-1. Владеет навыком проектировать и планировать построение систем искусственного интеллекта ПК-4.2. Н-2. Владеет способностью решать задачи с реальными данными так, чтобы попадать в первые 30% турнирной таблицы на сайте международных соревнований по машинному обучению kaggle.com

IV. СОДЕРЖАНИЕ И СТРУКТУРА ДИСЦИПЛИНЫ

Трудоемкость дисциплины составляет 5 зачетных единиц (180 часов), из них 34 часа лекционных занятий, 52 часа лабораторных занятий, 58 часов на самостоятельную работу и 36 часов на подготовку к экзамену.

Форма отчетности: экзамен

4.1 Содержание дисциплины, структурированное по темам, с указанием видов учебных занятий и отведенного на них количества академических часов

№ п/п	Раздел дисциплины/темы	Семестр	Виды учебной работы, включая самостоятельную работу обучающихся и трудоемкость (в часах)				Формы текущего контроля успеваемости Форма промежуточной аттестации *(по семестрам)*
			Контактная работа преподавателя с обучающимися			Самостоятельная работа	
			Лекции	Семинарские (практические занятия)	Лабораторные занятия		
1	**Раздел 1.** Алгоритмы машинного обучения. Язык программирования Python и библиотеки Pandas и Scikit-learn	3	1		2	50	Выполнение лабораторных работ
1.1	Основные определения: прецедент, обучающая выборка, признаки объектов, виды признаков, матрица объектов-признаков. Модель алгоритмов, метод обучения, функционал качества алгоритма. Вероятностная постановка задачи обучения. Принцип максимума правдоподобия. Связь максимизации правдоподобия и минимизации эмпирического риска. Проблема переобучения и обобщающая способность алгоритма. Состоятельные методы обучения. Эмпирические оценки обобщающей способности. Выбор алгоритма для вероятностной постановки задачи. Функционал среднего риска. Методы сбора и разметки данных	3	1		2	25	

№ п/п	Раздел дисциплины/темы	Семестр	Виды учебной работы, включая самостоятельную работу обучающихся и трудоемкость (в часах)				Формы текущего контроля успеваемости Форма промежуточной аттестации *(по семестрам)*
			Контактная работа преподавателя с обучающимися			Самостоятельная работа	
1.2	Байесовский подход к обучению. Приближенное вычисление плотности распределения. Наивный байесовский классификатор. Одномерный случай. Многомерный случай. Смеси распределений. EM-алгоритм разделения смеси. Смеси многомерных нормальных распределений.	3	1		2	10	
1.3	Метрические алгоритмы классификации. Обобщенный метрический классификатор. Виды и особенности частных случаев: методы ближайшего соседа, k ближайших соседей, взвешенных соседей, парзеновского окна постоянной и переменной ширины. Классификация объектов по значению отступа. Алгоритм STOLP отбора эталонных объектов. Выбор метрики и проклятие размерности.	3	1		2	5	
1.4	Линейные алгоритмы классификации. Модель Мак Каллока-Питтса, алгоритм стохастического градиента для минимизации функционала среднего риска. Эвристики для улучшения сходимости и обобщающей способности. Логистическая регрессия. Случайные величины с экспонентным законом распределения. Теорема о линейности байесовского классификатора.	3	1		2	5	
	Метод опорных векторов (SVM). Случай линейно разделимой выборки. Случай линейно неразделимой выборки. Функция Лагранжа. Классификация объектов в зависимости от значений множителей Лагранжа. Двойственная задача. Обучение SVM.	3	1		2	5	
	Кривая ошибок и AUC. Формула вычисления AUC. Примеры.	3	1		2	5	
	Алгоритмы восстановления регрессии. Метод наименьших квадратов. Непараметрическая регрессия. Многомерная линейная регрессия. Подход с использованием SVD-разложения матрицы. Гребневая регрессия. Метод главных компонент	3	1		2	5	

№ п/п	Раздел дисциплины/темы	Семестр	Виды учебной работы, включая самостоятельную работу обучающихся и трудоемкость (в часах)				Формы текущего контроля успеваемости Форма промежуточной аттестации *(по семестрам)*
			Контактная работа преподавателя с обучающимися			Самостоятельная работа	
	PCA.						
	Логические методы классификации. Понятие информативности предиката: эвристическое, вероятностное, энтропийное. Поиск информативных закономерностей. Алгоритмы для номинальных и порядковых признаков. Градиентный алгоритм синтеза коньюнкций. Построение решающего списка и решающего дерева. Редукция деревьев.	3	1		2		
	Композиции алгоритмов. Примеры: простое и взвешенное голосование, бэггинг, метод случайных подпространств.	3	1		2		
	Рекомендательные системы. Оценки качества. Алгоритмы построения рекомендательных систем.	3	1		2		
	Язык программирования Python. Библиотеки Pandas и Scikit learn	3	1		2	3	Выполнение лабораторных работ.
2	**Раздел 2.** Решение задач машинного обучения	3	1		2	50	Выполнение лабораторных работ.
2.1	Приближенное вычисление плотности распределения случайных величин. Построение предсказания на основе приближения	3	1		2	4	Выполнение лабораторных работ.
2.2	Метрические алгоритмы классификации	3	1		2	8	Выполнение лабораторных работ.
2.3	Логистическая регрессия	3	1		2	8	Выполнение лабораторных работ.
2.4	SVM	3	1		2	8	Выполнение лабораторных работ.
2.5	Бустинг	3	1		2	10	Выполнение лабораторных работ.
3	**Решение задач с реальными данными. Работа над проектом в группах по 2 чел.**	3			2	33	
	Подготовка к экзамену	3				36	

№ п/п	Раздел дисциплины/темы	Семестр	Виды учебной работы, включая самостоятельную работу обучающихся и трудоемкость (в часах)				Формы текущего контроля успеваемости Форма промежуточной аттестации *(по семестрам)*
			Контактная работа преподавателя с обучающимися			Самостоятельная работа	
Итого часов			**10**		**28**	**133**	

4.2 План внеаудиторной самостоятельной работы обучающихся по дисциплине

Семестр	Название раздела, темы	Самостоятельная работа обучающихся			Оценочное средство	Учебно-методическое обеспечение самостоятельной работы
		Вид самостоятельной работы	Сроки выполнения	Затраты времени (час.)		
1	Python	Выполнение дополнительных заданий	1 неделя	7	Проверка выполненных заданий	Материалы лекций, рекомендованная учебная литература, материалы, выложенные на интернет-странице курса на сайте edu.mmcs.sfedu.ru
1	Приближенное вычисление плотности распределения случайных величин. Построение предсказания на основе приближения.	Выполнение дополнительных заданий	1 неделя	7	Проверка выполненных заданий	Материалы лекций, рекомендованная учебная литература, материалы, выложенные на интернет-странице курса на сайте edu.mmcs.sfedu.ru
1	Метрические алгоритмы классификации.	Выполнение дополнительных заданий	2 недели	14	Проверка выполненных заданий	Материалы лекций, рекомендованная учебная литература, материалы, выложенные на интернет-странице курса на сайте edu.mmcs.sfedu.ru
1	Логистическая регрессия.	Выполнение дополнительных заданий	2 недели	14	Проверка выполненных заданий	Материалы лекций, рекомендованная учебная литература, материалы, выложенные на интернет-странице курса на сайте edu.mmcs.sfedu.ru

Семестр	Название раздела, темы	Самостоятельная работа обучающихся			Оценочное средство	Учебно-методическое обеспечение самостоятельной работы
		Вид самостоятельной работы	Сроки выполнения	Затраты времени (час.)		
1	SVM	Выполнение дополнительных заданий	2 недели	14	Проверка выполненных заданий	Материалы лекций, рекомендованная учебная литература, материалы, выложенные на интернет-странице курса на сайте edu.mmcs.sfedu.ru
1	Бустинг	Выполнение дополнительных заданий	2 недели	14	Проверка выполненных заданий	Материалы лекций, рекомендованная учебная литература, материалы, выложенные на интернет-странице курса на сайте edu.mmcs.sfedu.ru
1	Решение задач с реальными данными. Работа над проектом в группах по 2 чел	Выполнение дополнительных заданий	8 недель	56	Проверка выполненных заданий	Материалы лекций, рекомендованная учебная литература, материалы, выложенные на интернет-странице курса на сайте edu.mmcs.sfedu.ru
Подготовка к экзамену				**36**		
Общая трудоемкость самостоятельной работы по дисциплине (час)				**133**		
Бюджет времени самостоятельной работы, предусмотренный учебным планом для данной дисциплины (час)				**133**		

4.3 Содержание учебного материала

Раздел 1. Алгоритмы машинного обучения. Язык программирования Pyhton и библиотеки Pandas и Scikit learn.

Тема 1.1. Основные понятия

Основные определения: прецедент, обучающая выборка, признаки объектов, виды признаков, матрица объектов-признаков. Модель алгоритмов, метод обучения, функционал качества алгоритма. Вероятностная постановка задачи обучения. Принцип максимума правдоподобия. Связь максимизации правдоподобия и минимизации эмпирического риска. Проблема переобучения и обобщающая способность алгоритма. Состоятельные методы обучения. Эмпирические оценки обобщающей способности. Выбор алгоритма для вероятностной постановки задачи. Функционал среднего риска. Методы сбора и разметки данных.

Тема 1.2. Байесовский подход к обучению

Приближенное вычисление плотности распределения. Наивный байесовский классификатор. Одномерный случай. Многомерный случай. Смеси распределений. EM-алгоритм разделения смеси. Смеси многомерных нормальных распределений.

Тема 1.3. Метрические алгоритмы классификации

Обобщенный метрический классификатор. Виды и особенности частных случаев: методы ближайшего соседа, k ближайших соседей, взвешенных соседей, парзеновского окна постоянной и переменной ширины. Классификация объектов по значению отступа. Алгоритм STOLP отбора эталонных объектов. Выбор метрики и проклятие размерности.

Тема 1.4. Линейные алгоритмы классификации

Модель Мак Каллока-Питтса, алгоритм стохастического градиента для минимизации функционала среднего риска. Эвристики для улучшения сходимости и обобщающей способности. Логистическая регрессия. Случайные величины с экспонентным законом распределения. Теорема о линейности байесовского классификатора.

Тема 1.5. Метод опорных векторов (SVM)

Случай линейно разделимой выборки. Случай линейно неразделимой выборки. Функция Лагранжа. Классификация объектов в зависимости от значений множителей Лагранжа. Двойственная задача. Обучение SVM.

Тема 1.6. AUC

Кривая ошибок и AUC. Формула вычисления AUC. Примеры.

Тема 1.7. Алгоритмы восстановления регрессии

Метод наименьших квадратов. Непараметрическая регрессия. Многомерная линейная регрессия. Подход с использованием SVD-разложения матрицы. Гребневая регрессия. Метод главных компонент PCA.

Тема 1.8. Логические методы классификации

Понятие информативности предиката: эвристическое, вероятностное, энтропийное. Поиск информативных закономерностей. Алгоритмы для номинальных и порядковых признаков. Градиентный алгоритм синтеза коньюнкций. Построение решающего списка и решающего дерева. Редукция деревьев.

Тема 1.9. Композиции алгоритмов

Примеры: простое и взвешенное голосование, бэггинг, метод случайных подпространств.

Тема 1.10. Рекомендательные системы

Оценки качества. Алгоритмы построения рекомендательных систем.

Тема 1.11. Python

Язык программирования Python. Библиотеки Pandas и Scikit learn.

Раздел 2. Решение задач машинного обучения

Тема 2.1. Приближенное вычисление плотности распределения случайных величин. Построение предсказания на основе приближения.

Тема 2.2. Метрические алгоритмы классификации.

Тема 2.3. Логистическая регрессия.

Тема 2.4. SVM.

Тема 2.5. Бустинг.

Раздел 3. Решение задач с реальными данными. Работа над проектом в группах по 2 чел

Вопросы, выносимые на самостоятельное изучение

Отдельные задания в лабораторных работах. Решение задач с реальными данными. Работа над проектом в группах по 2 чел

V. ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

При проведении лекций и лабораторных занятий используются следующие образовательные технологии:

— мультимедийные лекции

— электронные формы контроля

— участие в международных конкурсах

Учебный процесс базируется на концепции компетентностного обучения, ориентированного на формирование конкретного перечня профессиональных компетенций, актуализацию получаемых теоретических знаний. Развертывание компетентностной модели обучения предполагает широкое применение инновационных способов организации учебного процесса, в т.ч. применение метода проектного обучения, технологий управляемого самостоятельного обучения в том числе балльно-рейтинговой системы, а также внедрение системы онлайн-поддержки внеаудиторной работы студентов.

VI. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ

Полный комплект контрольно-оценочных материалов (Фонд оценочных средств) оформляется в виде приложения к рабочей программе дисциплины.

VII. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

7.1. Основная литература.

Коэльо Л.П., Ричарт В. Построение систем машинного обучения на языке Python / Л.П. Коэльо, В. Ричарт; перевод с английского — 2-е изд. — Москва : ДМК Пресс, 2016. — 302 с. — ISBN 978-5-97060-330-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/82818?category_pk=1557&publisher=1028 (дата обращения: 10.10.2021). — Режим доступа: для авториз. пользователей.

7.2. Дополнительная литература.

[Электронный ресурс biblioclub: https://biblioclub.ru/index.php?page=book_red&id=576264] Гультяева Т. А. , Попов А. А. , Саутин А. С. Методы статистического обучения в задачах регрессии и классификации: монография / Изд. НГТУ 2015. - 323 с.

[Электронный ресурс biblioclub: http://biblioclub.ru/index.php?page=book_red&id=457464&sr=1] Осипов Г. С. Методы искусственного интеллекта / Москва: Физматлит, 2011. - 296 с

7.3. Список авторских методических разработок.

Нет

7.4. Периодические издания *(при необходимости)*

Нет

7.5. Перечень ресурсов сети Интернет, необходимых для освоения дисциплины

[Электронный ресурс http://www.machinelearning.ru/] Machinelearning.ru

[Электронный ресурс https://www.kaggle.com] Kaggle

Страница курса на сайте Института математики, механики и компьютерных наук, URL http://edu.mmcs.sfedu.ru/course/view.php?id=201

К.В. Воронцов Курс лекций по машинному обучению http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_(курс_лекций%2C_К.В.Воро%25

Система проведения международных соревнований по машинному обучению http://kaggle.com

7.6. Программное обеспечение информационно-коммуникационных технологий

Операционная система Linux, пакеты: Anaconda, Jupyter notebooks.

VIII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

8.1. Учебно-лабораторное оборудование

При проведении дисциплины учащиеся должны быть обеспечены:

1. Лекционной аудиторией с мультимедийным презентационным оборудованием для демонстрации презентаций и иллюстративного материала.

2. Аудиторией для лабораторных занятий с аппаратными и программными средствами в соответствии с реализуемой учебной тематикой.

8.2. Программные средства

Операционная система Linux, пакеты: Anaconda, Jupyter notebooks.

УЧЕБНАЯ КАРТА ДИСЦИПЛИНЫ

«Машинное обучение»

5 зач.ед.; ак.ч всего: 180, в т.ч.: 34 лекций, 52 лабор., 58 самостоят., 36 на подг. к экз.

Курс 1 год магистратуры, семестр 2

Направление подготовки: 02.04.02 «Прикладная математика и информатика»

№	Виды контрольных мероприятий	Текущий контроль	Рубежный контроль *(при наличии)*
	Раздел 1. Алгоритмы машинного обучения. Язык программирования Python и библиотеки Pandas и Scikit learn	**4**	
1.	Выполнение лабораторных работ	4	
	Раздел 2. Решение задач машинного обучения	**26**	
2.	Выполнение лабораторных работ	26	
	Раздел 3 Решение задач с реальными данными. Работа над проектом в группах по 2 чел		**30**
1.	Решение задач с реальными данными. Работа над проектом в группах по 2 чел		30
	Бонусные баллы	10 (за повышенную сложность выбранного задания)	
	Промежуточная аттестация в форме экзамена	40 письменная работа с демонстрацией работы алгоритмов	

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»
Институт магистратуры
Кафедра фундаментальной и прикладной математики

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

ПРИКЛАДНОЕ МАШИННОЕ ОБУЧЕНИЕ

Код и наименование направления подготовки/специальности:
01.04.02 «Прикладная математика и информатика»

Уровень образования:
Магистратура

Магистерская программа:
«Искусственный интеллект: математические модели и прикладные решения»

Форма обучения:
Очно-заочная

ПЕРЕЧЕНЬ КОМПЕТЕНЦИЙ, ФОРМИРУЕМЫХ ДИСЦИПЛИНОЙ

«Прикладное машинное обучение»

Код компетенции	Формулировка компетенции
1	2
ПК	**ПРОФЕССИОНАЛЬНЫЕ КОМПЕТЕНЦИИ**
ПК-3	Способен разрабатывать и применять методы и алгоритмы машинного обучения для решения задач
ПК-4	Способен руководить проектами по созданию комплексных систем искусственного интеллекта

ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ ПО ДИСЦИПЛИНЕ

Прикладное машинное обучение

№ п/п	*Контролируемые разделы дисциплины**	*Код контролируемой компетенции*	*Наименование оценочного средства***
1.	Алгоритмы машинного обучения. Язык программирования Python и библиотеки Pandas и Scikit learn	ПК-3.1. У-1, ПК-3.2. У-1, ПК-3.3. У-1, ПК-4.1. У-1, ПК-4.2. У-1	Выполнение лабораторных работ.
2.	Решение задач машинного обучения	ПК-3.1. У-1, ПК-3.2. У-1, ПК-3.3. У-1, ПК-4.1. У-1, ПК-4.2. У-1	Выполнение лабораторных работ
3.	Решение задач с реальными данными. Работа над проектом в группах по 2 чел	ПК-3.1. У-1, ПК-3.2. У-1, ПК-3.3. У-1, ПК-4.1. У-1, ПК-4.2. У-2	Работа над проектом в группах.
4.	Раздел 1. Алгоритмы машинного обучения. Язык программирования Python и библиотеки Pandas и Scikit learn Раздел 2. Решение задач машинного обучения	ПК-3.1. З-1, ПК-3.2. З-1, ПК-3.3. З-1, ПК-4.1. З-1, ПК-4.2. З-1, ПК-4.2. З-2	Экзамен

** Наименование раздела указывается в соответствии с рабочей программой дисциплины.*

***Наименование оценочного средства указывается в соответствии с учебной картой дисциплины.*

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»
Факультет компьютерных технологий и защиты информации
Кафедра фундаментальной и прикладной математики

Лабораторные работы

по дисциплине «Прикладное машинное обучение»

Лабораторная работа №1.

1. Запустите jupyter notebook, создайте новую тетрадку в папке с данными data.csv
2. Прочтите данные из файла data.csv.

```
import pandas as pd
data = pd.read_csv('data.csv', delimiter=',')
data.describe()   #вывод описания данных
```

3. Отобразите несколько первых записей

```
data.head(10)   #первые 10 строк
```

4. Прочтите в файле DataDictionary-ru.txt, что означают столбцы матрицы
5. Заметьте, что столбец «DebtRatio» содержит неправдоподобные данные. Только значения, соответствующие известному месячному доходу, являются отношениями. Остальные – абсолютные значения месячных выплат процентов. Исправьте данные, сделав все значения столбца «DebtRatio» абсолютными (умножьте их на MonthlyIncome). Чтобы ваша программа быстро работала на полных данных, постарайтесь не использовать цикл.

```
Обращение к элементам DataFrame:
  элемент: data.loc[i, 'названиеСтолбца']
  столбец: data['названиеСтолбца']
  подматрица: data.loc[a:b, списокНазванийСтолбцов]

Условная индексация:
data.loc[ data['столбец']>20, списокНазванийСтолбцов]
лучше писать так:
i =  data['столбец']>20  # вектор True и False
data.loc[i, 'названиеСтолбца']

У подматриц номера строк наследуются от исходной.

Циклы:
for i in range(0,20):
    оператор1
    оператор2
оператор после цикла

Итерация по строкам DataFrame:
for idx, row in data.iterrows():
    ...  idx - номер строки, row['столбец'] - значение элемента

pandas.isnull(скаляр или массив) - проверка, является ли значение неопределенным (NaN)
pandas.notnull(скаляр или массив) - проверка, является ли значение определенным (не NaN)
```

6. Поменяйте имя столбца на «Debt»: data = data.rename(columns={'DebtRatio' : 'Debt'})

7. Вычислите средний ежемесячный доход и присвойте всем клиентам с неизвестным доходом полученное число.

   data['столбец'].mean() - среднее, еще есть min, max, sum и др (см. http://pandas.pydata.org/pandas-docs/stable/api.html#dataframe)

8. (самостоятельная работа) Используя метод groupby, оцените вероятности невозврата кредита (SeriousDlqin2yrs=1) для различных значений количества иждивенцев (NumberOfDependents). Проделайте аналогичную процедуру для различных значений столбца NumberRealEstateLoansOrLines

   data['столбец1'].groupby(data['столбец2']).mean() - расчет средних значений столбца1 по гуппам из столбца2

9. (самостоятельная работа) Визуализируйте данные:
   a. Постройте график рассеяния на осях «age» и «Debt». Синим отметьте клиентов без серьезных задолженностей (SeriousDlqin2yrs = 0) и красным — должников: SeriousDlqin2yrs = 1.
   b. Постройте на одном графике две нормированные плотности распределения: красную – для месячного дохода клиентов с задолженностями, синюю – для месячного дохода клиентов без задолженностей. По оси абсцисс отобразите значения до 25000.

      Рисование:
      %matplotlib inline
      import matplotlib.pyplot as plt
      plt.plot(x, y) см. http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot
      plt.show()
      График рассеяния: plt.scatter(x,y) см. http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.scatter
      Рисование гистограмм: plt.hist см. http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.hist

      Рисование нескольких графиков на одном:
      fig, ax = plt.subplots()
      ax.hist(...)
      ax.hist(...)
      plt.show()

      Логарифмическая шкала:
      ax.set_xscale('log') или ax.set_yscale('log')

      Ограничение области графика:
      ax.axis([x1, x2, y1, y2])

Лабораторная работа №2

1. Загрузите данные о проданных на аукционе автомобилях. Целевым признаком здесь является наличие скрытых продавцом существенных недостатков. Посмотрите на данные.
2. Постройте график рассеяния с пробегом в качестве оси абсцисс и ценой MMRCurrentRetailAveragePrice в качестве ось ординат. Автомобили без недостатков отметьте зеленым цветом, со скрытыми недостатками — красным.
3. **Регрессия**
   a. Импортируйте из библиотеки sklearn все модели машинного обучения:
      from sklearn import *
      Трактуя задачу о предсказании наличия недостатков как задачу регрессии, натренируйте линейную модель LinearRegression на каком-нибудь подмножестве признаков (например, тех же VehOdo и MMRCurrentRetailAveragePrice):
      model1 = linear_model.LinearRegression()
      model1.fit(data.ix[:,['VehOdo','MMRCurrentRetailAveragePrice']], data['IsBadBuy'])
   b. Выполните предсказание для всех объектов обучающей выборки:
      prediction = model1.predict(data.ix[:,['VehOdo','MMRCurrentRetailAveragePrice']])

c. Преобразуйте вектор предсказаний prediction к значениям 0,1. Это можно сделать, например, так:
predictionClass = [1 if prediction[i]>0.5 else 0 for i in range(data.shape[0])]

d. Оцените качество классификации с помощью функций построения отчета classification_report и матрицы ошибок confusion_matrix:
print(metrics.classification_report(data['IsBadBuy'], predictionClass, target_names=['class0', 'class1']))
print(metrics.confusion_matrix(data['IsBadBuy'], predictionClass))

4. **Классификация**
   a. Трактуя задачу как задачу классификации, постройте модель классификации «решающее дерево» глубины 20: tree.DecisionTreeClassifier(max_depth=20) (все аналогично линейной регрессии).
   b. Решающее дерево предсказывает вероятности. С помощью построенной модели рассчитайте вероятности наличия скрываемых недостатков:
   predictionProb =
   model2.predict_proba(data.ix[:,['VehOdo','MMRCurrentRetailAveragePrice']])
   c. Преобразуйте вероятности к одному вектору из предсказаний 0,1. Это можно сделать, например, так: predictionClass = [0 if predictionProb[i][0]>0.5 else 1 for i in range(data.shape[0])]
   Постройте отчет о классификации и матрицу ошибок. Какой метод оказался лучше?
   d. Объясните в комментариях каждое число в отчете классификации с точки зрения матрицы ошибок. Напишите арифметические действия с коэффициентами матрицы ошибок, которые приводят к каждому числу отчета из столбцов precision(точность) и recall(полнота).
5. **Кроссвалидация**
   a. Разделите исходную выборку на две части:
   dataTrain = data.ix[0:34999,]
   dataTest = data.ix[35000:69999,]
   b. Натренируйте решающее дерево на dataTrain и примените к dataTrain и dataTest, вычислив для каждого случая точность предсказания:
   metrics.accuracy_score(dataTrain['IsBadBuy'], predictionClass)
   Проведите несколько экспериментов для различных глубин дерева. Напишите в комментариях, для каких глубин модель недообучена, для каких переобучена и где точка раннего останова.
6. (самостоятельная работа) **Решающая функция (Decision function)**
   a. Вернемся к модели решающего дерева глубины 20, построенного по всей обучающей выборке data. Пусть цена ошибки неправильного предсказания 0 равна 1000, а неправильного предсказания 1 — 100:
   lossTrue0Pred1 = 100
   lossTrue1Pred0 = 1000
   Вычислите функцию потерь - среднюю ошибку на всей обучающей выборке.
   b. Выясните, как нужно изменить решающую функцию:
   [0 if predictionProb[i][0]>0.5 else 1 for i in range(data.shape[0])]
   чтобы функция потерь была минимальна? Найдите оптимальную решающую функцию и минимальное значение функционала потерь. (Вам поможет формула из лекции 2)

Лабораторная работа №3.

1. Загрузите данные о погибших в катастрофе с Титаником. Целевым признаком здесь является факт спасения пассажира. Посмотрите на данные. Для простоты исключите пассажиров с неизвестным возрастом: data = data.dropna(subset=['Age']).reset_index(drop=True)
2. Постройте график рассеяния с возрастом в качестве оси абсцисс и ценой билета в качестве оси ординат. Отметьте красным — погибших, зеленым — спасшихся пассажиров. Постройте отдельно 2 графика: для женщин и для мужчин. Выполняется ли для данного множества гипотеза компактности?
3. Запрограммируйте функцию вычисления расстояния между пассажирами:

```
def distance(a,b):
    d = 0
    d += abs(a['Pclass'] - b['Pclass'])
    d += a['Sex'] != b['Sex']
    d += abs(a['Age'] - b['Age'])
    d += abs(a['SibSp'] - b['SibSp'])
    d += abs(a['Parch'] - b['Parch'])
    d += abs(a['Fare'] - b['Fare'])
    d += a['Embarked'] != b['Embarked']
    return d
```

4. Метод KNeighborsClassifier из библиотеки sklearn излишне требователен к типам данных. Запрограммируем метод k ближайших соседей вручную:

```
import numpy as np
def myKNeighborsClassifier(learnData, K, passengerIndexForPrediction):
    dists = np.zeros((learnData.shape[0]-1,2))
    i = 0
    for idx, row in learnData.iterrows():
        if idx != passengerIndexForPrediction:  #LOO метод контроля ошибки
            dists[i][0] = distance(learnData.ix[passengerIndexForPrediction,], row)
            dists[i][1] = row['Survived']
            i += 1
    dists = sorted(dists, key = lambda pair: pair[0])
    prediction = 0
    for i in range(K): prediction += dists[i][1]
    prediction /= K
    return round(prediction)
```

5. Запустите предсказание для всех пассажиров судна и, используя LOO-контроль ошибки оцените качество классификации:

```
accuracy = 0
for idx, row in data.iterrows():
    accuracy += row['Survived'] == myKNeighborsClassifier(data, 5, idx)
print(accuracy/data.shape[0])
```

6. (самостоятельная работа) Подберите слагаемые в метрике, их веса и количество соседей так, чтобы качество классификации было максимальным. Если функция на python работает долго, воспользуйтесь ее аналогом на
   C++: KNeighborsClassifier.cpp. Компиляция на Linux выполняется командой
   g++ -O2 KNeighborsClassifier.cpp -o KNeighborsClassifier
   Запуск: ./KNeighborsClassifier
   Построившему метрику с наилучшим качеством: +5 бонусных баллов
7. (самостоятельная работа) Создайте копию функции myKNeighborsClassifier и измените ее так, чтобы она считала выступ объекта. Назовите ее calcMargin. Вычислите выступы для всех объектов и выведете информацию о пяти объектах с минимальный выступом (шумовых выбросах) и пяти объектах с максимальным выступом (эталонах). Объясните (в комментариях) полученные данные. Это легче всего сделать с помощью функции np.argsort, возвращающей индексы элементов в отсортированном ряду:
   idx = np.argsort(margins)

```
print('Белые вороны (аномалии)')
print(data.loc[idx[:5],])
print('Эталоны (характерные объекты)')
print(data.loc[idx[-5:],])
```

Лабораторная работа №4

7. Данными в этом задании являются измерения некоторых проверочных параметров на конвейерах сборки оборудования Bosh (см. конкурс «Bosch Production Line Performance» на Kaggle). Все исходные данные Bosh не помещаются в оперативную память компьютера, поэтому в файле data.csv — лишь некоторые признаки. Прочтите данные из файла data.csv. Целевым признаком здесь является Responce — наличие брака в оборудовании.
8. Постройте на одном графике два приближения к плотности распределения признака L1_S24_F1846 для Responce=0 и для Response=1, используя одно из следующих ядер (номер ядра выберите по формуле: (n mod 6)+1, где n — ваш номер в списке группы):
    1. кусочно-постоянное (прямоугольное) - tophat
    2. гауссовское - gaussian
    3. линейное (треугольник) - linear
    4. косинусоидальное - cosine
    5. квадратичное (Епанечникова) - epanechnikov
    6. экспоненциальное - exponential

   Help:
```
from sklearn.neighbors.kde import KernelDensity
import numpy as np
i0 = data['Response']==0
kde0 = KernelDensity(kernel='gaussian', bandwidth=0.1).fit(data.ix[i0,'L1_S24_F1846'].values.reshape(-1, 1))
X_plot = np.linspace(-1, 1, 1000).values.reshape(-1, 1)
Dens0 = np.exp(kde0.score_samples(X_plot))  #score_samples возвращает логарифм плотности
fig, ax = plt.subplots()
ax.plot(X_plot, Dens0, color='blue')
```

   Ответьте в комментариях на вопрос: является ли выборка хорошо разделимой по признаку L1_S24_F1846 ?
9. Разбейте выборку data на две равные части: тренировочную dataTrain и проверочную dataTest. Пользуясь кроссвалидацией, подберите для каждого класса Response значение ширины ядра bandwidth, при котором логарифм правдоподобия максимален на проверочной выборке:
```
dataTrain = data.ix[0:data.shape[0]/2,].reset_index()
dataTest = data.ix[data.shape[0]/2:data.shape[0],].reset_index()
r = 0
kde0 = KernelDensity(kernel='gaussian', bandwidth=0.1).fit(dataTrain.ix[dataTrain['Response']==r,'L1_S24_F1846'].values.reshape(-1, 1))
print(kde0.score_samples(dataTest.ix[dataTest['Response']==r,'L1_S24_F1846'].values.reshape(-1, 1)).sum())
```
10. (самостоятельная работа) Для найденных наилучших bandwidth с помощью команд:
    predictionProbXafter0 =np.exp(kde0.score_samples(dataTest['L1_S24_F1846'].values.reshape(-1, 1)))
    predictionProbXafter1 =np.exp(kde1.score_samples(dataTest['L1_S24_F1846'].values.reshape(-1, 1)))
    вычислите p(x|0) и p(x|1) для тестовой выборки. По формуле Байеса найдите затем p(0|x) и p(1|x). Отсортируйте все объекты тестовой выборки по возрастанию предсказанной вероятности p(1|x), выведете на экран вероятности для последних 10 объектов и рассчитайте количество бракованных деталей среди последних 100 объектов в отсортированном ряду.
    Help: сортировка, возвращающая индексы элементов: ind = np.argsort(predictionProb1afterX)
    печать вероятностей для последних 10: print(predictionProb1afterX[ind[-10:-1]])
    количество бракованных среди100 с максимальной предсказанной вероятностью брака:
    print(sum(dataTest.ix[ind[-100:-1],'Response']))

Лабораторная работа №5

1. Рассмотрим данные медицинской страховой фирмы. На основе характеристик пациентов Members.csv (возраст, пол) и данных о получении медицинского обслуживания в предыдущем году Claims_Y1.csv (медицинское учреждение, врач, тип проблемы, количество дней госпитализации, дата, и др.) нужно предсказать факт госпитализации хотя бы на 1 день в следующем году DaysInHospital_Y2.csv.

```
import pandas as pd
from sklearn import *
%matplotlib inline
import matplotlib.pyplot as plt
days2 = pd.read_csv('DaysInHospital_Y2.csv', index_col='MemberID')
m = pd.read_csv('Members.csv', index_col='MemberID')
claims = pd.read_csv('Claims_Y1.csv')
```

2. Чтобы анонимизировать данные организатор указал приблизительную информацию о пациентах, например в столбце возраст указаны возрастные группы: '0-9', '10-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80+'. Преобразуем строковые признаки в количественные и заменим пропущенные значения:

```
i = pd.notnull(m.AgeAtFirstClaim)
m.ix[i,'AgeAtFirstClaim'] = m.ix[i,'AgeAtFirstClaim'].apply(lambda s: s.split('-')[0] if s!='80+' else '80')
m.ix[i,'AgeAtFirstClaim'] = m.ix[i,'AgeAtFirstClaim'].apply(lambda s: int(s))
m.AgeAtFirstClaim = m.AgeAtFirstClaim.fillna(value=-1)
m.Sex = m.Sex.fillna(value='N')
claims.CharlsonIndex = claims.CharlsonIndex.map({'0':0, '1-2':1, '3-4':3, '5+':5})
claims.LengthOfStay = claims.LengthOfStay.fillna(value=0)
claims.LengthOfStay = claims.LengthOfStay.map({0:0, '1 day':1, '2 days':2, '3 days':3, '4 days':4,\
    '5 days':5, '6 days':6, '1- 2 weeks':10, '2- 4 weeks':21, '4- 8 weeks':42, '26+ weeks':182})
```

3. Сконструируем признаки по массиву случаев медицинского обслуживания:
   f_Charlson — максимальный индекс коморбидности Чальсона по всем случаям для пациента,
   f_LengthOfStay — суммарное количество дней госпитализации в прошлом году:

```
f_Charlson = claims.groupby(['MemberID']).CharlsonIndex.max()
f_LengthOfStay = claims.groupby(['MemberID']).LengthOfStay.sum()
```

4. Составим матрицу объектов признаков со столбцами: f_Charlson, f_LengthOfStay, возраст пациента, ClaimsTruncated (не оказалось ли случаев медицинского обслуживания слишком много):

```
data = days2
data = data.join(f_Charlson)
data = data.join(f_LengthOfStay)
data = data.join(m['AgeAtFirstClaim'])
data.head(5)
```

5. Составим функцию, которая будет делить выборку на две части dataTrain и dataTest, обучать логистическую регрессию на dataTrain, применять к dataTest, строить кривую ошибок и считать под ней площадь:

```
def calcAUC(data):
    dataTrain, dataTest = cross_validation.train_test_split(data, test_size = 0.5, random_state=1)
    model = linear_model.LogisticRegression()
    model.fit( dataTrain.ix[:, dataTrain.columns != 'DaysInHospital'], dataTrain.DaysInHospital )
    predictionProb = model.predict_proba( dataTest.ix[:, dataTest.columns != 'DaysInHospital'] )
    fpr, tpr, _ = metrics.roc_curve(dataTest['DaysInHospital'], predictionProb[:,1])
    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.show()
    print( metrics.roc_auc_score(dataTest['DaysInHospital'], predictionProb[:,1]) )
```

6. Применим данную функцию к data: calcAUC(data)
7. Логистическая регрессия принимает на вход только количественные признаки. Добавим к нашим данным пол пациента, применив one hot encoding:

```
pol = pd.get_dummies(m.Sex, prefix='pol')
data2 = data.join(pol)
calcAUC(data2)
```

8. (самостоятельная работа) Попытайтесь применить one hot encoding к уже существующим в data2 признакам или составить новые признаки по массиву claims. Построившему матрицу объектов признаков, для которой логистическая регрессия работает с наилучшим качеством: +5 бонусных баллов.

Лабораторная работа №6

1. Данная задача относится к области обработки естественного языка (Natural language processing). Требуется научить компьютер понимать текст и предсказывать отношение (Sentiment) автора к описываемой теме. Данные в файле recs.txt представляют собой отзывы о фильмах с сайта kinopoisk.ru.
2. Распакуйте архив и прочтите данные из файла:
```
import pandas as pd
import numpy as np
import nltk
import re
from sklearn import *
pd.set_option('max_colwidth',600)
d0 = pd.read_csv('recs.txt', sep="\-\-\|\|\|\-\-", encoding='utf-8')
d0.head(3)
```
3. В базе данных — 100 тысяч отзывов. Каждый описывается полями: Sentiment (отношение), URL (адрес на сайте) и Text. Преобразуем текст в набор признаков методом «bag of words»: каждому слову будет соответствовать бинарный признак — встречается это слово в тексте или нет. Для этого сначала профильтруем текст, оставив только значащие слова. Скачайте из корпуса набор незначащих слов для фильтрации командой nltk.download() (в появившемся окне вкладка Corpora, пункт stopwords).
4. Импортируйте список незначащих слов:
```
from nltk.corpus import stopwords
stopwords = set(stopwords.words("russian"))
print(stopwords)
```
5. Определите функцию фильтрации текста:
```
def review_to_words( review_text ):
    review_text = review_text.replace('ё','е')
    review_text = review_text.replace('Ё','Е')
    # 1. Удаляем все, кроме букв
    letters_only = re.sub("[^а-яА-Я]", " ", review_text)
    # 2. Делаем все буквы строчными и создаем массив слов
    words = letters_only.lower().split()
    # 3. Удаляем незначащие слова
    meaningful_words = [w for w in words if not w in stopwords]
    # 4. Формируем текст, объединяя слова через пробел
    return( " ".join( meaningful_words ))
```
6. Профильтруйте все отзывы:
```
d1 = d0.copy()
d1['Text'] = d1['Text'].apply(lambda s: review_to_words(s))
d1.head(3)
```
7. Запустите преобразователь текстового признака в «bag of words»:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer = "word", tokenizer = None, preprocessor = None, stop_words = None,\
                             max_features = 50000, binary = True)
bagOfWords = vectorizer.fit_transform(d1['Text'])
vocabulary = vectorizer.get_feature_names()
```
8. Теперь в наших матрицах объектов-признаков 50 тысяч столбцов и в сумме 100 тысяч строк. Если 1 элемент занимает 1 байт, то суммарный объем будет 5 Гб. Проверьте по диспетчеру задач (системный монитор в Linux), сколько занимает памяти ваша программа и напишите в комментариях. Все ли согласуется?
9. Разбейте выборку на две части: обучающую и проверочную
```
bagOfWordsTrain = bagOfWords[0:50000,:]
bagOfWordsTest = bagOfWords[50000:100000,:]
```
10. Натренируйте машину опорных векторов на обучающей выборке, примените к проверочной и вычислите процент правильных предсказаний:
```
SVM = svm.LinearSVC(C=1, fit_intercept=False)
SVM.fit(bagOfWordsTrain, d1.iloc[0:50000]['Sentiment'])
SVM.score(bagOfWordsTest, d1.iloc[50000:100000]['Sentiment'])
```
11. Подберите важность суммы нарушений C, чтобы SVM работала лучше всего
12. Выведите на экран 20 слов, которые больше всего голосуют за позитивность отзыва и 20 — за негативность вместе с соответствующими им коэффициентами натренированной линейной функции:
```
ind = np.argsort(SVM.coef_[0])
for i in range(20):
    print(SVM.coef_[0][ind[i]], vocabulary[ind[i]])
print("\n")
```

```
for i in range(20):
    print(SVM.coef_[0][ind[-i-1]], vocabulary[ind[-i-1]])
```

13. Вычислите выступы, найдите и выведете на экран отзывы с минимальным и максимальным выступом, величину выступа, величину предсказания и сентиментность отзыва. Объясните в комментариях, почему найденный отзыв имеет такой маленький выступ, а другой найденный — такой большой:

```
prediction = SVM.decision_function(bagOfWordsTest)
margins = np.multiply(prediction, d1.iloc[50000:100000]['Sentiment'].as_matrix())
ind = np.argsort(margins)
i = ind[-1] + 50000
print(margins[ind[-1]], prediction[ind[-1]], d0.ix[i,'Sentiment'], d0.ix[i,'Text'])
print("\n")
i = ind[0] + 50000
print(margins[ind[0]], prediction[ind[0]], d0.ix[i,'Sentiment'], d0.ix[i,'Text'])
```

14. (самостоятельная работа) Для объяснения отступов вам возможно пригодится код, который печатает коэффициенты w_i линейной функции, значения соответствующих признаков и соответствующие слова. И все это делается в порядке убывания модуля произведения $w_i x_i$:

```
k = ind[0]
ind2 = np.argsort(np.abs(np.multiply(SVM.coef_[0], bagOfWordsTest[k].todense()))).reshape(-1,1)
s = 0
for i in range((bagOfWordsTest[k]!=0).sum()):
    ii = ind2[-i-1].item()
    print('wi='+str(SVM.coef_[0][ii]), 'xi='+str(bagOfWordsTest[k,ii]),  vocabulary[ii])
    s += SVM.coef_[0][ii]*bagOfWordsTest[k,ii]
print('Проверка:',s)
```

15. (самостоятельная работа) Нормализуйте признаки перед разбиением выборки на Train и Test так, чтобы вместо 0/1 стояли частоты употребления слов, деленные на длину отзыва. Подберите наилучшую константу C и опишите, как изменятся отзывы с минимальным и максимальным выступом? Что лучше: не нормализованные или нормализованные признаки? Почему?

```
Нужно установить binary = False в CountVectorizer и добавить после fit_transform
bagOfWords = preprocessing.normalize(bagOfWords, norm='l1')
```

Лабораторная работа №7

1. Попробуем применить самые сильные на сегодняшний день композиции логических алгоритмов к рассмотренной в лабораторной работе 5 задаче о медицинском страховании Heritage Provider Network Health Prize. Загрузите подготовленные данные с множеством дополнительных признаков. По заданному набору обращений клиентов в больницу для каждого клиента были рассчитаны количества обращений со всеми Specialty, PrimaryConditionGroup и ProcedureGroup. Загрузите данные:

```
import pandas as pd
from sklearn import *
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from catboost import CatBoostClassifier

data = pd.read_csv('data_HPN_best.csv');
dTrain, dTest, YTrain, YTest = model_selection.train_test_split(data.ix[:, data.columns!='DaysInHospital'], data['DaysInHospital'], test_size=0.5,
random_state=1)
data.head(5)
```

2. Победитель конкурса в лабораторной работе 5 с помощью логистической регрессии достиг AUC = 0.7112. Используя расширенный набор данных и мощные алгоритмы RandomForest и CatBoost от Yandex, улучшите его рекорд. Необходимо научиться настраивать алгоритмы так, чтобы оба показали результат лучше упомянутого. Для сравнения выведете на экран лучшую точность работы одного классифицирующего дерева:

```
Tree = tree.DecisionTreeClassifier()
Tree.fit(dTrain, YTrain)
aucTreeTrain = metrics.roc_auc_score(YTrain, Tree.predict_proba(dTrain)[:,1])
aucTreeTest = metrics.roc_auc_score(YTest, Tree.predict_proba(dTest)[:,1])
print('             Train    Test')
print('Tree:        {:6.4}  {:6.4}'.format(aucTreeTrain, aucTreeTest))
```

```
RandomForest = ensemble.RandomForestClassifier()
RandomForest.fit(dTrain, YTrain)
aucRFTrain = metrics.roc_auc_score(YTrain, RandomForest.predict_proba(dTrain)[:,1])
aucRFTest = metrics.roc_auc_score(YTest, RandomForest.predict_proba(dTest)[:,1])
print('Random forest: {:6.4}  {:6.4}'.format(aucRFTrain, aucRFTest))

CatBoost = CatBoostClassifier()
CatBoost.fit(dTrain, YTrain)
accRFTrain = metrics.roc_auc_score(YTrain, CatBoost.predict_proba(dTrain)[:,1])
accRFTest = metrics.roc_auc_score(YTest, CatBoost.predict_proba(dTest)[:,1])
print('CatBoost:      {:6.4}  {:6.4}'.format(accRFTrain, accRFTest))
```

3. (самостоятельная работа) Найдите 10 самых информативных признаков:

```
fi = np.array(CatBoost.get_feature_importance(dTrain,YTrain))
m = len(fi)
ind = np.argsort(fi)
cols = list(reversed(dTrain.columns[ind[m-10:m+1]]))
fi = list(reversed(fi[ind[m-10:m+1]]))

from pylab import rcParams
rcParams['figure.figsize'] = 12,7
fig = plt.figure()
plt.rc('font', family='Arial')
plt.rcParams['xtick.labelsize'] = 20
plt.bar(np.arange(len(cols)), fi, color='g')
plt.xticks(np.arange(len(cols))+0.4, cols, rotation='vertical')
fig.suptitle('Важность признаков', fontsize=20)
```

Критерии оценки:

За каждую выполненную лабораторную работу студент получает одинаковое количество баллов, равное 30/7.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»
Факультет компьютерных технологий и защиты информации
Кафедра фундаментальной и прикладной математики

Групповой проект
по дисциплине **Прикладное машинное обучение**

Студенты образуют команды 2 человек и выбирают задачи, размещенные в сети интернет на платформе kaggle.com проведения соревнований по машинному обучению (англоязычный ресурс).

Пример задания

Deloitte/FIDE Chess Rating Challenge

Наборы данных для этого конкурса включают реальные исторические данные, предоставленные Всемирной шахматной Федерацией (ФИДЕ). Участники конкурса будут тренировать свои рейтинговые системы, используя обучающий набор данных из более чем 1,84 миллиона игровых результатов для более чем 54 000 шахматистов за последние одиннадцать лет. Затем участники используют свой метод, чтобы предсказать исход еще 100 000 игр, сыгранных среди тех же игроков в те-

чение следующих трех месяцев. Конкурсные работы будут оцениваться автоматически веб-сайтом на основе точности их прогнозов. Две работы в день могут быть представлены каждой командой, и призы будут определены в соответствии с лучшей оценкой каждой команды.

Спонсор конкурса, Deloitte Australia, предоставил приз в размере $ 10,000 команде, которая представляет наиболее точные прогнозы. "Делойт" является выдающимся аналитическим поставщиком во всем мире и помогает компаниям собирать, управлять и анализировать свои данные в рамках общей бизнес-стратегии.

Она пожертвовала шахматные программы с подписями известных игроков на три команды, занявшие 2-е/3-е/4-е место.

Кроме того, представители ФИДЕ вручат специальный "приз ФИДЕ" за то, что они считают наиболее перспективным подходом, из десяти наиболее точных записей, которые соответствуют ограничительному определению "практической шахматной рейтинговой системы". Это ограничительное определение указано в правилах конкурса. Для победителя ФИДЕ предоставит тариф на перелет туда и обратно в Афины, Греция, и полный пансион на три ночи в Афинах, а также оплату других расходов, чтобы один человек представил и обсудил свою систему во время специальной встречи экспертов по шахматам в Афинах.

Рейтинговая система Elo была разработана венгерским физиком Арпадом Эло в 1950-х годах и принята Всемирной шахматной Федерацией (ФИДЕ) в 1970 году. На протяжении более четырех десятилетий система FIDE Elo служила основным мерилом в мире для измерения силы шахматистов. Рейтинги ФИДЕ используются для определения приглашений на шахматные турниры, включая цикл чемпионата мира, расчета конкретных пар в большинстве шахматных турниров и предоставления титулов, таких как Международный Мастер (IM) или гроссмейстер (GM). На самом деле система Elo настолько популярна, что она была адаптирована ко многим другим приложениям, помимо шахмат, включая рейтинги командных видов спорта, другие настольные игры и онлайн-системы видеоигр.

Однако, несмотря на популярность системы Эло, никогда не было продемонстрировано, что подход Эло технически превосходит другие подходы. Большая часть привлекательности системы Elo связана с ее простотой и знакомством, и она идеально подходила для того времени, когда расчет рейтингов был значительной практической задачей даже для ежегодного списка из нескольких сотен игроков. Формула эло была выведена теоретически, в эпоху без больших объемов исторических данных или чего-либо приближающегося к сегодняшней вычислительной мощности. Благодаря мощным компьютерам и большим игровым базам данных мы можем легко исследовать подходы, которые могут быть лучше, чем Elo, при прогнозировании результатов шахмат. Такое исследование может иметь серьезные последствия для теории и практики рейтинговой методологии, как для шахмат, так и для мира за пределами шахмат.

В качестве первого шага в этом процессе Каггле осенью 2010 года провел конкурс "Эло против остального мира", требуя от участников разработать прогностические модели, которые могли бы с большой точностью прогнозировать результаты шахматных игр. Конкурс был очень популярен как среди любителей шахмат, так и среди ученых-данных, привлекая более 3500 заявок от 258 команд-участников из 41 страны. Хотя призовой фонд был минимальным, было огромное конкурентное стремление, побуждающее многих участников к постоянным усилиям. Например, несмотря на ограничение, удерживающее команды от подачи более двух заявок в день, было двадцать команд, которые сделали более 50 заявок на протяжении всего конкурса. Базовые данные настолько просты и понятны, что очень легко начать играть с моделями прогнозирования, но общая проблема максимизации точности настолько сложна, что даже масштабные усилия первого конкурса не были достаточными для определения явно превосходящего подхода. Было большое разнообразие в методологиях только десяти лучших призеров, все из которых задокументировали свои подходы в значи-

тельной степени после завершения конкурса. Эталонная подача формулы Эло закончилась далеко, далеко, позади, на 141-м месте из 258, и было 39 команд, чьи прогнозы были по крайней мере на 5% более точными, чем система Эло. Понятно, что система Elo не самая точная, но остается неясным, какая система превосходит ее.

Это уже не вопрос "Эло против остального мира"; теперь мы должны провести второй конкурс и сосредоточиться на поиске подходящей замены Эло. Возможно, лучшим подходом будет модификация подхода Эло, или, возможно, это будет что-то совершенно новое. Это второй конкурс будет иметь несколько существенных улучшений по сравнению с первым конкурсом. Он будет иметь более надежную функцию подсчета очков, выбранную после обширного анализа результатов первого конкурса, а также консультации с мировыми лидерами в теории шахматного рейтинга и категориального анализа данных. Кроме того, ФИДЕ предоставил полный набор данных о многолетних результатах отдельных игр, которые использовались для расчета официальных рейтингов ФИДЕ. До сих пор это никогда не было возможно, чтобы выполнить этот анализ, потому что набор данных не был собран, и этот конкурс сайт является единственным местом в мире, где имеются данные второго конкурса обеспечивает более чем в 30 раз, как много игр, как первый конкурс все-таки, и гораздо большее население шахматистов, а также, отражающие весь распределении сила игрока, а не просто доля топ-игроков, охваченных первый конкурс.

Конкурсные работы будут содержать прогнозируемый Белый счет в каждой из 100 000 игр в тестовом наборе. Прогнозы для отдельных игр будут оцениваться отдельно, а совокупное "отклонение" будет рассчитываться как среднее отклонение во всех играх. Также отметим, что несколько тысяч игр являются фиктивными (фейковыми) матчапами, включенными с целью отбить у участников "майнинг" тестового набора для получения дополнительной информации о силе каждого игрока. Эти игры игнорируются функцией оценки.

Выигрышная одиночная подача будет иметь минимальное Биномиальное отклонение. Легко заметить, что прогнозирование ожидаемой оценки 0% или 100% имеет неопределенное Биномиальное отклонение, поскольку LOG10 (0) Не определен. Кроме того, даже если белые выигрывают, маржинальная выгода для прогноза чуть выше 99% минимальна по сравнению с прогнозом 99%. И точно так же, даже если черные выигрывают, маржинальная выгода для предсказания чуть ниже 1% минимальна по сравнению с предсказанием 1%. Поэтому нет никаких веских причин предсказывать ожидаемый результат выше 99% или ниже 1%. Таким образом, для целей подсчета очков, все прогнозы белого балла выше 99% будут рассматриваться как 99%, а все прогнозы белого балла ниже 1% будут рассматриваться как 1%. Вы можете" ограничить " свои прогнозы на 1% и 99% самостоятельно, или вы можете позволить функции подсчета очков сделать это за вас; в любом случае Ваш счет будет таким же.

Тестовый набор данных списков 100,000 игр во время тестового периода (месяца 133-135). Каждая из этих игр однозначно идентифицируется TEID (Test Game ID) в диапазоне от 1 до 100000. Тестовый набор данных предоставляет достаточную информацию о каждой из этих игр, включая месяц, в котором она была сыграна, и идентификационные номера для белых и черных, чтобы вы могли предсказать исход каждой игры.

При отправке прогнозов необходимо включить файл со всеми 100 000 играми, а также строку заголовка. Для каждой игры вы указываете TEID и ожидаемую оценку для белого (десятичное значение от 0 до 1). Обратите внимание, что функция оценки будет рассматривать все ожидаемые оценки ниже 0.01 в 0.01, и будет рассматривать все ожидаемые оценки выше 0.99 0.99 а.

Вы должны убедиться, что ваши игры перечислены в порядке возрастания, в соответствии с TEID. Первая строка-это строка заголовка, вторая строка-это ваш прогноз на TEID#1, третья строка-это ваш прогноз на TEID#2, и так далее, заканчивая последний ряд, имеющий свой прогноз на TEID#100000. Поэтому все файлы должны иметь ровно 100001 строк.

Файлы должны быть отформатированы в формате с Разделителями-запятыми (CSV), имея ".CSV-файла" расширения и разделения значений в одной строке через запятую.

В качестве примера, если вы представили файл, содержащий прогнозы 0,500 для ожидаемого результата белых в первых трех играх, первые четыре строки будут выглядеть следующим образом:

TEID, WhiteExpect
1,0.500000
2,0.500000
3,0.500000

Файлы данных охватывают последние 135-месячные результаты профессиональных шахматных игр, извлеченные из базы данных Всемирной шахматной Федерации (ФИДЕ). Период обучения составляет 132 месяца, за которым следует трехмесячный период тестирования (133-135 месяцев). Конкретных сроков не предусмотрено; все игры просто целое число месяц ИД# от 1 до 135. Есть 54,205 различных шахматистов, включенных в данные; эти игроки однозначно идентифицируются по идентификатору в диапазоне от 1 до 54,205.

В рамках конкурса предоставляется восемь различных файлов:

(1-3) для обучения вашей системы прогнозирования можно использовать набор данных начального обучения (включая 1840 124 игры). Он разделен на три отдельных файла: часть 1 (месяцы 1-96), часть 2 (месяцы 97-118) и часть 3 (месяцы 119-132).

(4-5) вторичный набор данных для обучения (в том числе игры 312,511) и высшего учебного набора данных (включая игры 265,577) обеспечить дополнительный обучающих данных, которые могут быть полезны для проверки вашей системы и прогнозирования результатов игры. Для участия в конкурсе не требуется использовать эти дополнительные обучающие наборы данных, но большинство систем значительно выиграют от использования вторичных и третичных обучающих наборов данных. Более подробную информацию можно найти на странице дополнительные наборы данных обучения.

(6) тестовый набор данных (включая 100 000 игр) идентифицирует шахматные партии в течение тестового периода, который должен быть предсказан. Также обратите внимание, что пример файла отправки приведен на странице инструкции по отправке.

(7) первоначальный рейтинговый список (включая 14,118 игроков) содержит первоначальный список рейтингов ФИДЕ игроков, входящих в начало тренировочного периода. Это составляет приблизительно 25% от общего количества игроков; начальные оценки для остальных 75% игроков должны быть рассчитаны на основе их начальных игр в течение тренировочного периода. Помните, что любой желающий претендовать на призовую категорию ФИДЕ должен использовать этот список, чтобы определить начальные рейтинги этих 14,118 игроков.

(8) пример файла представления (включая 100,001 строк) форматируется соответствующим образом для представления на конкурс, содержащий прогноз ожидаемого балла 50% для всех 100 000 игр в тестовом наборе. Это идентично тесту всех ничьих. Он включает в себя строку заголовка, за которой следуют прогнозы для тестовых игр #1, #2, #3, ..., #99,998, #99,999, #100,000, в общей сложности 100,001 строк. Ваши фактические представления должны быть точно такими же, как этот файл, за исключением того, что он будет содержать ваши фактические прогнозы для каждой игры, а не универсальный 50%.

Набор Данных Начального Обучения

Есть 1,840,124 игры в наборе начальной подготовки, охватывающих последние одиннадцать-летний период (известный как месяц #1 через месяц #132). Они составляют 35-40% от почти пяти миллионов игровых результатов, которые были использованы ФИДЕ (Всемирной шахматной Федерацией) для расчета рейтингов Игроков за это время. Многие из этих пяти миллионов игровых результатов не доступны по отдельности в компьютерных файлах, и поэтому они должны были быть частично восстановлены из других источников для этого конкурса; именно поэтому в наборе начальной подготовки есть "только" 1,84 миллиона игр. На протяжении большей части тренировочного периода основной тренировочный набор содержит около 25% полного набора игр FIDE, используемых для официальных рейтинговых расчетов. Процент намного выше в последние два или три года тренировочного периода, благодаря недавним изменениям в правилах ФИДЕ о том, как организаторы турнира должны представлять результаты своего турнира ФИДЕ. Последние данные гораздо более доступны.

Есть три возможных исхода завершенной игры в шахматы - либо белые победы, либо игра обращается, или черные победы. Эти соответствуют "забить" на белом 1.0, 0.5, 0.0 баллов или, и наоборот, результат в черный 0.0, 0.5, 1.0 или очков. Турнир забил как правило, добавляет игрока в общем рейтинге, так что тот, кто выигрывает три игры, ничьи в пяти матчах, и проигрывает одну игру бы общий балл 5.5 из 9 игр, и (несмотря на меньшее количество побед) будет финишировать впереди кого-то, кто побеждает в пяти играх, рисует ноль игры и проигрывает четыре игры (общий балл 5.0 из 9). При этом важно учитывать не только то, сколько игр удается выиграть игроку, но и то, сколько потерь ему удается избежать. Белые всегда двигаются первыми, и поэтому есть небольшое преимущество в том, что белые фигуры (более выраженные среди сильнейших игроков). Например, в первичном тренировочном наборе 125 000 игр, где абсолютная разница в рейтингах ФИДЕ между игроками была меньше 20 очков, а средний балл белых в этих играх составил 53,7%.

Набор данных первичного обучения включает в себя 1840 124 игр, каждой из которых присваивается уникальный идентификатор первичной обучающей игры# (называемый PTID) от 1 до 1840 124. Для каждой игры, PTID, и месяц ИД (от 1 до 132), а игрока id #'s для белых и черных игроков, а также исход игры, от белого зрения (очки либо 1.0, 0.5, или 0.0). Рейтинги fide игроков не представлены в наборе данных первичной тренировки; часть задачи для вас, чтобы сделать свои собственные оценки изменения сильных сторон игроков с течением времени.

Наконец, набор данных основного обучения предоставляет два избыточных значения (WhitePlayerPrev и BlackPlayerPrev) для каждого результата игры, указывая количество игр (в наборе основного обучения), сыгранных каждым из двух игроков в течение предыдущих 24 месяцев. Эти значения могут быть вычислены путем просмотра данных за предыдущие месяцы в наборе данных начального обучения, а не путем явного предоставления, но для Вашего удобства они были явно предоставлены для каждой обучающей игры.

Почему WhitePlayerPrev и BlackPlayerPrev актуальны? Ну, пул активных игроков всегда увеличивается в размерах. Таким образом, в любой данный месяц, многие из игр, которые играют вовлечь игроков, которые ранее играли очень небольшое количество игр (или нет вообще). Было бы несправедливо включать многие из этих игр в набор тестов, поскольку прогнозы были бы совершенно неопределенными. Так при создании тестового набора за 133-135 месяцев был применен фильтр. Вместо того, чтобы включать все игры, сыгранные в течение месяцев 133-135, тестовый набор данных включает только игры, в которых оба игрока играли по крайней мере 12 игр в течение последних 24 месяцев основного набора данных обучения (т. е. месяцев 109-132). Участники, возможно, пожелают провести перекрестную проверку, опираясь на образцы из основного набора учебных материалов, которые будут иметь схожие характеристики с тестовым набором данных, с тем чтобы разработать модель или оптимизировать параметры в модели. Для создания собственных проверочных наборов из обучающих данных можно применить тот же фильтр выборки, который использовался для создания тестового набора (требуется WhitePlayerPrev > 12 и BlackPlayerPrev > 12). Включение значений WhitePlayerPrev и BlackPlayerPrev в тренировочные игры позволяет это сделать.

Например, если ваши проверки испытания была взята из игры на месяц 130 только, и только некоторые игры с этого месяца возникли WhitePlayerPrev > 12 и BlackPlayerPrev > 12, то вы бы только быть игры, в которых оба игрока как минимум 12 игр в течение предыдущих 24 месяцев первичная подготовка набора (т. е. месяцев 106-129). И этот фильтр аналогичен тому, что было сделано для фильтрации игр для тестового набора.

Тестовый Набор Данных

Тестовый набор данных (охватывающий 100 000 игр между месяцами 133-135) очень похож на основной обучающий набор данных, хотя он намеренно опускает столбец WhiteScore. Конечно, смысл конкурса заключается в том, чтобы участники точно предсказывали WhiteScore для всех игр в тестовом наборе данных, поэтому тестовый набор данных указывает только личности игроков и месяц, в котором игра была сыграна, и не сообщает вам результаты игры. Тестовый набор данных включает все известные игры, сыгранные в 133-135 месяцах, где оба игрока сыграли 12 или более игр в последние 24 месяца набора данных начальной подготовки.

Набор тестовых данных включает 100 000 игр, каждой из которых присваивается уникальный идентификатор тестовой игры# (называемый TEID) от 1 до 100 000. Для каждой игры, TEID, и месяц ИД (между 133 и 135), и ID игрока за белых и черных игроков, не предусмотрено. TEID особенно важен, поскольку он будет использоваться для уникальной идентификации каждой тестовой игры при отправке прогнозов и для сортировки строк в файле отправки.

Обратите внимание, что Вы не должны использовать тестовый набор данных в качестве дополнительного источника подсказок о силе игрока. Прогнозы на месяцы 133-135 должны основываться на оценочных игровых способностях игроков в конце месяца 132, и эти прогнозы должны быть полностью перспективными, как если бы вы сделали прогнозы прямо в конце месяца 132. Мы могли бы попросить полный кросс-продукт (54,205 x 54,205 x 3) предсказаний всех возможных игр, а не только 100 000 игр из тестового набора, но это было бы слишком большим файлом для отправки. Вместо этого, чтобы препятствовать участникам "майнинга" тестового набора данных для получения дополнительной информации, тестовый набор данных из 100 000 игр включает в себя много тысяч ложных совпадений. Прогнозы для этих конкретных игр будут отброшены и не будут использоваться при подсчете очков. Метод генерации ложных совпадений не является случайным и не раскрывается. Ложные игры намеренно включаются, чтобы побудить людей прогнозировать только перспективно.

Критерии оценки:

Каждый из студентов группы получает 30 баллов при условии попадания команды в top-30% рейтинговой таблицы. Если у группы возникают трудности, преподаватель помогает, чтобы указанный рейтинг был достигнут. Данный курс читался в магистратуре мехмата ЮФУ уже более 5 лет. За все его историю еще не было студентов, которые не смогли бы преодолеть барьер top-30%.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»
Факультет компьютерных технологий и защиты информации
Кафедра фундаментальной и прикладной математики

Вопросы к экзамену

Прикладное машинное обучение

(наименование дисциплины)

Перечислите альтернативные названия области науки Машинное обучение (хотя бы 2 штуки) (0,5 балла)

Можно ли использовать машинное обучение, чтобы автоматически оценивать сочинения школьников? Если да, то что для этого нужно, опишите математическую постановку задачи машинного обучения. Если нет, объясните почему. (1 балл)

Перечислите типы задач машинного обучения. Укажите какие типы можно свести к другим и каким образом. (2 балла)

Перечислите типы признаков объектов в задачах машинного обучения. Чем они друг от друга отличаются? (1 балл)

Что математически означают термины модель и алгоритм обучения модели? (1 балл)

Опишите самый популярный общий алгоритм обучения модели. (1 балл)

Перечислите проблемы, из-за которых переходят к вероятностной постановке задачи машинного обучения. (0.5 балла)

Каким образом происходит обучение в вероятностной постановке задачи машинного обучения в случае, когда нужно найти плотность вероятности и функция потерь не задана. (0.5 балла)

Что такое решающая функция (decision function)? Где и как она применяется? По каком правилу она работает? (1 балл)

Для заданного объекта x и классов y in {-1,+1} алгоритм машинного обучения рассчитал условные вероятности p(x|y): p(x|+1)=0.8, p(x|-1)=0.4. Какой класс нужно предсказать объекту x, чтобы вероятность ошибки была минимальна, если в обучающей выборке 30% объектов имеют класс y=+1 и 70% класс y=-1. Если оба решения имеют одинаковые вероятности ошибки, так и напишите. Обоснуйте решение математически. (2 балла)

Банку нужно принять решение о выдаче клиенту кредита величиной 1 млн. руб. на 1 год под 15% годовых. В случае отрицательного решения банк рискует потерять сумму, равную 15% от величины кредита. А в случае положительного решения и невозврата кредита клиентом банк рискует потерять сумму, равную величине кредита. Для заданного клиента x и классов y in {вернет, не вернет} алгоритм машинного обучения рассчитал условные вероятности p(y|x): p(вернет|x)=0.9, p(не вернет|x)=0.1. Помогите банку принять верное решение, минимизирующее величину среднего риска. Если оба решения приводят к одинаковому среднему риску, так и напишите. Обоснуйте решение математически. (2.5 балла)

Задана выборка одномерной случайной величины: X^ell={4,1,6,2,4,3,2,5,9,4}. Постройте приближение к плотности распределения, используя параметрический подход и нормальное распределение. (2 балла)

В обучающей выборке 60% объектов имеют класс y=+1 и 40% класс y=-1. Параметрический подход рассчитал приближенные частные плотности распределения p(x_1|y) и p(x_2|y) признаков x_1 и x_2 для каждого класса y\in\{-1,+1\}:
p(x_1|+1) \sim N(5,1), p(x_1|-1) \sim N(6,1), p(x_2|+1) \sim N(-2,1), \ p(x_2|-1) \sim N(-3,1).
Здесь N(a,\sigma^2) - нормальное распределение с матожиданием a и дисперсией \sigma^2. Используя наивный байесовский подход, напишите, как вы будете рассчитывать вероятности классов y=\pm 1 для объекта (x_1,x_2) = (7,-1.5). Приближенные вычисления экспонент выполнять не нужно, пусть они останутся в формуле ответа как есть. (2 балла)

Плотность вероятности p(x_1,x_2|y) распределения объекта \bar x=(x_1,x_2) в классах y=\pm 1 есть

$$p(x_1, x_2|+1) = \frac{1}{2\pi} e^{-\frac{1}{2}((x_1-3)^2+(x_2-1)^2)},$$
$$p(x_1, x_2|-1) = \frac{1}{2\pi} e^{-\frac{1}{2}((x_1-2)^2+(x_2-2)^2)}.$$

Проверьте выполнение условий теоремы о логистической регрессии (напишите эту проверку!) и, если они выполняются, найдите уравнение линии, разделяющей классы, и выразите вероятности p(y|x_1,x_2) через логистическую функцию. (3 балла)

Напишите формулу и нарисуйте на графике зависимость функции потерь от выступа объекта для логистической регрессии. Нарисуйте на этом же графике бинарную функцию потерь. (1 балл)

Задана матрица объектов-признаков:

x_1	x_2	y
да	дождь	5
да	без осадков	-7
нет	снег	-1
да	без осадков	-5
нет	без осадков	-8

да & снег & -2
нет & снег & 0
нет & без осадков & -3
да & без осадков & -10

Примените бинаризацию (one hot encoding) к признаку x_2. Напишите получившуюся матрицу объектов-признаков. (2 балла)

Опишите геометрический смысл метода наименьших квадратов. (2 балла)

При каких условиях метод наименьших квадратов совпадает с принципом максимума правдоподобия? (2 балла)

Чем гребневая регрессия отличается от обычного метода наименьших квадратов? Напишите формулы. (1 балл)

Напишите вывод решения метода наименьших квадратов через компоненты SVD-разложения матрицы объектов-признаков. (3 балла)

Как число обусловленности матрицы связано с ее сингулярными значениями? Напишите формулу (1 балл)

Из-за чего возникают проблемы в применении метода наименьших квадратов к обучающей выборке с мультиколлинеарными признаками? Как с ними борются? (2 балла)

Что такое и как работает метод главных компонент (PCA)? (2 балла)

Как SVD-разложение связано с методом главных компонент (PCA)? Напишите условие теоремы. (2 балла)

Критерии оценки:
Каждый студент получает свой набор заданий из указанных выше, сгенерированный авторской программой распределения вариантов. Программа следит за тем, чтобы суммарная сложность получающихся билетов у всех студентов составляла 40 баллов, а также за распределением вопросов по темам, рассадкой студентов в классе (вопросы распределяются так, чтобы рядом сидящие студенты получали различные задания). За каждую правильно отвеченный вопрос студенту начисляются баллы указанные в скобках после формулировки вопроса.