МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет прикладной математики, информатики и механики

Кафедра математических методов исследования операций

Отчет

о прохождении учебной практики по получению первичных профессиональных умений и навыков научно-исследовательской деятельности

Направление 01.03.02 - Прикладная математика и информатика

Сроки прохождения практики: 14.07.2020 – 27.07.2020

Обучающийся _____ С. С. Ноздрин
Руководитель _____ д.т.н., проф. Ю. В. Бондаренко

Воронеж - 2020

Содержание

1. Введение

Проведение научного исследования чаще всего заключается в выявлении скрытых правил и закономерностей в наборах данных, формулировке гипотез и выявлении типовых структур. Для этого приходится использовать различные методы обнаружения знаний. Человеческий разум не приспособлен для восприятия больших массивов разнородной информации. Для расширения аналитических возможностей человека можно использовать различные методы анализа данных.

В рамках учебной практики по получению первичных профессиональных умений и навыков научно-исследовательской деятельности была рассмотрена задача, целью которой является исследование рынка труда Российскоф Федерации, а также построения модели машинного обучения для предсказания ежемесячной заработной платы граждан на основе имеющихся признаков.

Информационной базой исследования являются данные НИУ ВШЭ взятые за 2018 год.

Общий объем выборки составляет 18954 респондентов, в ходе исследования респонденты с большим процентом пропущенных значений не учитываются, а также не идут в рассмотрение признаки с большим количеством пропусков.

В конечном исследовании участие принимают 7734 респондента. В качестве среды обработки данных используется дистрибутив языков программирования Python и R - Anaconda, а также среда программирования Jupyter Notebook со встроенным языком Python.

В ходе данного исследования использовались различные методы анализа данных для отбора признаков, их сокращения, формирования новых, а также были построены модели машинного обучения для решения предиктивной задачи.

2. Теоретическая часть

Для решения поставленных в исследовании задач, применялись различные методы анализа данных и машинного обучения.

Так, использовался факторный анализ для сокращения числа переменных и преобразования их в факторы.

В качестве моделей машинного обучения использовались нейронные сети и градиентный бустинг из пакета xgboost.

Перейдем к теоретическому описанию данных методов.

2.1. Факторный анализ

Факторный анализ является важным инструментом для решения практических задач при разработке рекомендательных систем (и не только), который используется прежде всего для редукции данных и определения структуры взаимосвязи данных.

Факторный анализ переживал несколько волн развития и какое-то время был объявлен "плохой"математикой(как, например, и нейронные сети в первые этапы развития) в первую очередь за высокий уровень субъективности и интуиции при анализе результатов, а также "пренебрежением"к некоторым математическим понятиям(например, к вопросу о высоком или невысоком уровне скоррелированности полученных факторов и исходных признаков)

Задачи, которые решает факторный анализ:

1) Сокращение числа переменных

Факторный анализ позволяет сократить число переменных в n-ое количество раз и ,как мы надеемся, сохранить всю или важную информацию о данных, однако это невсегда возможно.

Этот инструмент позволяет объединить переменные, которые обладают общим свойством и в какой-то степени похожи(обладают высокой корреляцией), в один фактор, измеряющие то же, что и исходные переменные.

2)Измерение "неизмеримого"

Часто мы имеем дело с косвенными признаками, которые объясняют что-то одно, но с разных сторон. Факторный анализ позволяет объединить эти

внешние признаки в один и измерить это свойство.

3)Выявление структуры зависимости данных

В данном контексте факторный анализ позволяет описать структуру данных прежде всего через призму корреляционной зависимости между признаками.

Например, если несколько переменных имеют сильную корреляцию, то они зависимы, а если слабую, то нет. Поэтому "зависимые"переменные стоит объединить в один фактор, а переменные, которые не имеют высоких корреляций с другими переменными, стоит оставить, т.к они объясняют что-то своё(содержат уникальную информацию).

Обращу внимание, что здесь и проявляется один из моментов, почему факторный анализ называют "плохой"математикой, т.к. здесь происходит пренебрежение математическими понятиями о корреляции, т.е о наивном предположении, например, что низкий уровень корреляции указывает на отстутствие зависимости между признаками, что совершенно не так(т.к можно отрицать лишь линейную зависимость).

4)Проецирование данных

Факторный анализ позволяет проецировать многомерные данные на двумерную/трехмерную плоскость. Задача заключается в отыскании лучшей проекции, которая позволит описать данные наилучшим образом.

Однако все зависит от того, что мы хотим увидеть:

a)Сократить число переменных и сохранить "расстояния"между исходными примерами

b)Найти такую проекцию, которая лучше разделит несколько "облаков"точек (используется при задачах класификации/кластеризации)

Также это важный пункт для улучшения визуализации данных. Например, существует алгоритм t-SNE(как пакет в языке Python,например), который позволяет сократить размерность данных и визуализировать их.

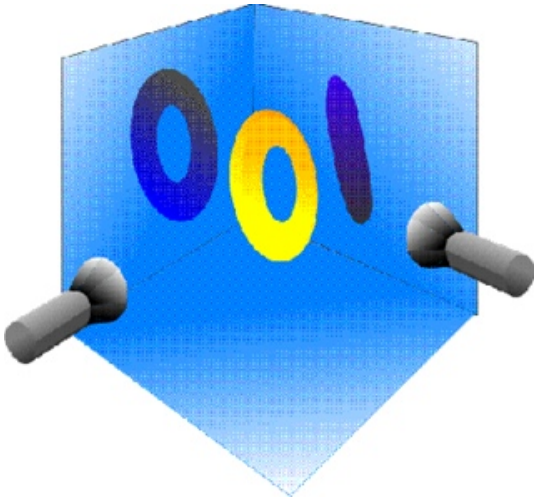

Рис.1. Пример поиска наилучшей проекции

5)Преодоление мультиколленарности

Часто в случае многомерных задач, когда требуется построить предиктивную модель, мы сталкиваемся с проблемой большого количества скоррелированных данных, что влияет на качество модели. Когда таких признаков очень много, то "отбрасывание"лишних превращается в мучительный процесс. На помощь приходит факторный анализ, который позволяет найти такие факторы, которые будут ортогональны между собой в пространстве и будут "независимы".Правда, существует издержка, в виде потери ясности интерпритации полученных признаков и их влияния на модель.

Когда заходит разговор о факторном анализе, существует небольшая путаница.

Дело в том, что факторным анализом называют как классический факторный анализ, так и метод главных компонент.

Метод главных компонент наиболее популярен сегодня, он фактически проводится, когда применяется SVD-разложение, поэтому он и использовался при решении данной задачи.

Итак, пусть у нас есть вектор признаков $X = (X_1, ..., X_k)$, где k - количество признаков.

Задача состоит в том, чтобы найти наиболее информативную проекцию, наибольшое облоко точек, т.е точки с наибольшим разбросом. Как измерить степень разброса? Будем это делать с помощью дисперсии.

Итак, будем искать компоненту(фактор) в виде линейной комбинации признаков с наибольшей дисперсией.

$Y_1 = a_{11}X_1 + a_{12}X_2 + .. + a_{1k}X_k$

$D[Y_1] \to max$

Так как дисперсия зависит от параметров вектора $\vec{a}$, то введем дополнительное условие: $a_i a_j = 1$, где $a_i = (a_{i1}, .., a_{ik})$

Итак, мы построили первый фактор, теперь нужно построить второй таким же образом:

$Y_2 = a_{21}X_1 + a_{22}X_2 + .. + a_{2k}X_k$, но этот фактор должен быть ортогонален в пространстве признаков первому, то есть мы должны потребовать независимость, т.е нескоррелированость этих двух компонент, то есть:

$D[Y_2] \to max$

$corr(Y_1, Y_2) = 0$

Тогда для k-ой компоненты получаем задачу:

$Y_k = a_{k1}X_1 + a_{k2}X_2 + .. + a_{kk}X_k$

$D[Y_k] \to max$

$corr(Y_k, Y_1) = 0$

$corr(Y_k, Y_2) = 0$

...

$corr(Y_k, Y_{k-1}) = 0$

Нахождение параметров **a** сводится к решению задачи:

$Ra = \lambda a$, где R - матрица ковариаций(корреляций) вектора $\mathbf{X}$, а λ - собственное значение

Можно работать с матрицей ковариацией, но тогда для сохранения интерпритации следует стандартизировать данные(т.е перейти к корреляциям)

Решив задачу из линейной алгебры $|R - \lambda I = 0|$, получим k собственных векторов, а также, что $D[Y] = \lambda$

То есть задача поиска компоненты с максимальной дисперсией сводится к поиску максимальному собственному значению

Также сделаем вывод, что $\sum \lambda_i = trace(R)$

Так как R - матрица корреляций, то получаем $\sum \lambda_i = k$

Расположив λ_i в порядке убывания, выберем то количество собственных чисел(факторов), которое нужно в конкретной задаче в зависимости от того, какое количество информации они "покроют".

Существует негласное правило: стоит отбросить все факторы, собственные значения которых меньше 0.8, но оно необязательное.

На практике удобно построить график каменистой осыпи.

2.2. Нейронные сети

Нейронные сети в последнее время обрели наибольшую популярность в методах машинного обучения, особенно при работе с изображениями и в задачах обработки естественного языка.

Однако этот метод не является самым сильным при решении задач с табличными данными.

В рамках нашего исследования нейронные сети используются для сравнения с другими методами машинного обучения, в частности с xgboost.

Итак, нейронная сеть - модель, представляющая собой совокупность нейронных слоев и весов, функции активации, метода оптимизации и функции потерь.

Нейронные слои представляют собой входы модели с предыдущего слоя. Самый первый (входной) слой есть признаки, затем количество в нейронов в слоях может увеличиваться или уменьшаться в зависимости от архитектуры.

Последний(активационный слой) представляет собой выход модели из нейронной сети, т.е результат её работы.

В задачах классификации на него принято навешивать функцию Softmax, в задачах же регрессии (в нашем случае) используется линейная функция.

Функция активации - функция, вычисляющая выходной сигнал искусственного нейрона. Существует множество функций активаций, но в последнее время обрела популярность функция ReLU и всевозможные её модификации.

$ReLU(x) = max(0, x)$

Функция потерь используется для нахождения оптимальных значений параметров (весов) нейронной сети.

Для задач регрессии используют, как правило, в качестве функции потерь средний квадрат ошибки, в то время как в задачах классификации - Кросс-Энтропию.

Отметим, что функция потерь должна быть дифференцируема в большинстве точек области определения, а также не иметь точек разрыва, иначе это может привести к затуханию градиентов.

Также существует интересная взаимосвязь функций активаций, функции потерь и проблемы затухания градиентов, именно поэтому функция активации

ReLU обрела наибольшую популярность сегодня.

Метод оптимизации используется для нахождения минимума функции потерь. Различных методов существует множество, но самые популярные - Adam и SGD с Nesterov Momentum.

2.3. Градиентный бустинг (xgboost)

XGBoost— одна из самых популярных и эффективных реализаций алгоритма градиентного бустинга на деревьях.

В основе XGBoost лежит алгоритм градиентного бустинга деревьев решений. Градиентный бустинг — это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений. Обучение ансамбля проводится последовательно в отличие, например от бэггинга. На каждой итерации вычисляются отклонения предсказаний уже обученного ансамбля на обучающей выборке. Следующая модель, которая будет добавлена в ансамбль будет предсказывать эти отклонения. Таким образом, добавив предсказания нового дерева к предсказаниям обученного ансамбля мы можем уменьшить среднее отклонение модели, котрое является таргетом оптимизационной задачи. Новые деревья добавляются в ансамбль до тех пор, пока ошибка уменьшается, либо пока не выполняется одно из правил "ранней остановки".

Данная модель имеет несколько гиперпараметров, которые необходимо подбирать для каждой конкретной задачи заново:

nestimators — число деревьев

eta — размер шага обучения,

gamma — минимальное изменение значения loss функции для разделения листа на поддеревья

maxdepth — максимальная глубина дерева

lambda/alpha — L2/L1 регуляризация.

3. Практическая часть

Исследование состоит из нескольких логических частей и написания нескольких скриптов, каждый из которых выполняет свою задачу.

С целью лучшего понимания хода работы, он будет поделен на несколько подчастей с подробным описанием.

3.1. Обработка данных

Так как данные представляют собой тип, родственный для работы в SPSS, нужна специальная подготовка для обработки в языке Python.

	psu	popul	v_age	v_diplom	vh5	vi1	vi3	vj6	vj6.1a	vj6.2	...	vm11311c	vj360.2	vj360.5
0	Волосовский р-н: Ленинградская область	12161.0	44.0	законченное высшее образование и выше	ЖЕНСКИЙ	В ДРУГОМ НАСЕЛЕННОМ ПУНКТЕ	В ГОРОДЕ	Нет	10	50	...	NaN	Да	Нет
1	Волосовский р-н: Ленинградская область	12161.0	62.0	законченное среднее образование	ЖЕНСКИЙ	В ДРУГОМ НАСЕЛЕННОМ ПУНКТЕ	В СЕЛЕ, ДЕРЕВНЕ, КИШЛАКЕ, АУЛЕ	Нет	24	48	...	NaN	Нет	Да
2	Волосовский р-н: Ленинградская область	12161.0	33.0	незаконченное среднее образование (7 - 8 кл) +...	ЖЕНСКИЙ	В ТОМ, ГДЕ ЖИВЕТ СЕЙЧАС	NaN	Нет	12	36	...	NaN	Нет	Нет
3	Волосовский р-н: Ленинградская область	12161.0	31.0	незаконченное среднее образование (7 - 8 кл) +...	МУЖСКОЙ	В ТОМ, ГДЕ ЖИВЕТ СЕЙЧАС	NaN	Нет	12	48	...	NaN	Да	Нет
4	Волосовский р-н: Ленинградская область	12161.0	73.0	незаконченное среднее образование (7 - 8 кл)	ЖЕНСКИЙ	В ДРУГОМ НАСЕЛЕННОМ ПУНКТЕ	В ГОРОДЕ	NaN	NaN	NaN	...	NaN	Нет	Нет
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18949	Волосовский р-н: Ленинградская область	2422.0	0.0	NaN	МУЖСКОЙ	В ТОМ, ГДЕ ЖИВЕТ СЕЙЧАС	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN

Рис.2. Фрагмент исходной таблицы данных

Для проведения более качественного анализа и построения модели машинного обучения, были отобраны важные (на взгляд исследователя) признаки, удалены строки с пустыми значениями в зависимом столбце и переименованы исходные признаки для лучшего понимания их сути.

	region	population	age	diploma	sex	move	birthlocation	subordinates	hours-per-day	hours-per-week	...	fitness	smartphone
0	Волосовский р-н: Ленинградская область	12161.0	44.0	законченное высшее образование и выше	ЖЕНСКИЙ	В ДРУГОМ НАСЕЛЕННОМ ПУНКТЕ	В ГОРОДЕ	Нет	10	50	...	NaN	Да
1	Волосовский р-н: Ленинградская область	12161.0	62.0	законченное среднее образование	ЖЕНСКИЙ	В ДРУГОМ НАСЕЛЕННОМ ПУНКТЕ	В СЕЛЕ, ДЕРЕВНЕ, КИШЛАКЕ, АУЛЕ	Нет	24	48	...	NaN	Нет
2	Волосовский р-н: Ленинградская область	12161.0	33.0	незаконченное среднее образование (7 - 8 кл) +...	ЖЕНСКИЙ	В ТОМ, ГДЕ ЖИВЕТ СЕЙЧАС	NaN	Нет	12	36	...	NaN	Нет
3	Волосовский р-н: Ленинградская область	12161.0	31.0	незаконченное среднее образование (7 - 8 кл) +...	МУЖСКОЙ	В ТОМ, ГДЕ ЖИВЕТ СЕЙЧАС	NaN	Нет	12	48	...	NaN	Да
4	Волосовский р-н: Ленинградская область	12161.0	73.0	незаконченное среднее образование (7 - 8 кл)	ЖЕНСКИЙ	В ДРУГОМ НАСЕЛЕННОМ ПУНКТЕ	В ГОРОДЕ	NaN	NaN	NaN	...	NaN	Нет
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18949	Волосовский р-н: Ленинградская область	2422.0	0.0	NaN	МУЖСКОЙ	В ТОМ, ГДЕ ЖИВЕТ СЕЙЧАС	NaN	NaN	NaN	NaN	...	NaN	NaN

Рис.3. Фрагмент измененной таблицы данных

3.2. Анализ данных

В этой части происходит анализ имеющихся признаков, их фильтрация, отбор и создания новых.

Так как их было довольно много и анализ представлял собой построение гистограмм распределения зависимой переменной(дохода за месяц) относительно значений признака, а также выдвижение гипотез и.т.д., выполнение данного пункта задачи будет рассмотренно на примере нескольких признаков.

Соотношение зарплаты мужчин и женщин в современном обществе вызывает массу дискуссий, поэтому возникла идея о рассмотрении данного признака и возможного его включения как переменной для построения модели машинного обучения.

Гистограммы зарплат оказались распределены по-разному, с небольшой ассиметрией и большим количеством выбросов, поэтому было принято решение для проверки гипотезы о равенстве зарплат, использовать тест из непараметрической статистики - критерией Манна-Уитни-Вилкоксона для независимых переменных.

```
: print('Male median salary:{}'.format(M['salarym'].median()))
  print('Female median salary:{}'.format(W['salarym'].median()))

  Male median salary:26500.0
  Female median salary:18000.0

: from scipy import stats
  stats.mannwhitneyu(M['salarym'],W['salarym'])

: MannwhitneyuResult(statistic=4800210.5, pvalue=3.277131539367278e-162)
```

Рис.4. Применение теста для исследования значимости пола для дохода

P-value оказался близким к 0, т.е гораздо меньше принятых в статистическом анализе "пороговых"значений (0.01,0.05,0.1), поэтому основная гипотеза о равенстве доходов мужчин и женщин была отвергнута, т.е существует статистически значимое различие в доходах представлителей разных полов.

Ещё одним примером проведеннного анализа может послужить изучение зависимости влияние уровня образования на размер заработной платы.

Изначально данные представляли из себя нескольких категорий, где-то различных, а где-то схожих между собой по степени влияния на заработную плату.

	diploma	edumedian:
0	законченное высшее образование и выше	25600.0
1	законченное среднее образование	20000.0
2	законченное среднее специальное образование	20000.0
3	незаконченное среднее образование (7 - 8 кл)	18000.0
4	незаконченное среднее образование (7 - 8 кл) +...	20000.0
5	окончил 0 - 6 классов	14000.0

Рис.5. Уровень образования и медианной заработной платы

Так как представителей образования с уровнем 0-6 классов очень мало, а также существует группы, где уровень образования и уровень заработной платы похож, были произведены преобразования данного признака следующим образом: начальный уровень образования 0-6 классов, а также незаконченное среднее, средний уровень - среднее образование, высший уровень - высшее образование.

Это позволило построить более различимые по медианной заработной плате группы образование, что тоже является полезным признаком.

	diploma	edumedian:
0	average	20000.0
1	high	25600.0
2	start	18000.0

Рис.6. Преобразованный уровень образования и медианной заработной платы

Таким образом, производился анализ каждого признака, на основе чего происходили его преобразования, построение новых (в основном методом dummy - кодирования), а также удаление лишних.

Примером лишнего признака, например, являлся ответ на вопрос о понижении зарплаты. Исследования показали, что статистически значимых отклонений относительно медианного уровня заработной платы нет, а отсутствие ответа не является важным, т.к. это произошло только с одним респондентом.

	decreasesalary	edumedian:
0		23000.0
1	Да	20000.0
2	ЗАТРУДНЯЮСЬ ОТВЕТИТЬ	20000.0
3	НЕТ ОТВЕТА	21500.0
4	Нет	21000.0
5	ОТКАЗ ОТ ОТВЕТА	3300.0

Рис.7. Снижение заработной платы и медианная заработная плата

Таким образом, в дальнейший анализ и построение моделей машинного обучения были отобраны следующие признаки: населенность города, пол респондента, количество рабочих часов в день и неделю, количество подчиненных, государственное предприятие, иностранное предприятие, уровень ощущения счастья, количество детей, частота занятия фитнесом, уровень владения иностранным языком, наличие смартфона, ноутбука, планшета, преобразованння ступень образования и локация рождения.

Лишними же в рамках данного анализа оказались сведения о снижении зарплаты, боязни потери работы, наличия переезда, окончание курсов, а также наличие отпуска и его продолжительность.

3.3. Подготовка данных

Прежде чем перейти к факторному анализу и построению моделей машинного обучения, необходимо было проделать небольшую подготовительную работу, которая включала в себя разделения выборки на матрицу признаков(X) и зависимую переменную y(заработная плата за предыдующий месяц).

Затем было произведено разделение на обучающую и тестовую выборку, для этого использовалась простейшная функция traintestsplit с заданным параметром testsize = 0.2, которая в случайном порядке распределяет данные на обучающую и тестовую выборку в соотношении 80 на 20 процентов.

Полученные выборки не должны были иметь смещений относительно друг друга, а также быть стратифицированны.

Тест Манна-Уитни-Вилкоксона показал довольно высокое значение p-value в рамках гипотезы о схожести распределения зарплат из обучающей и тестовой выборки, поэтому мы делаем вывод о том, что статистически они принадлежат одной совокупности.

```
from scipy import stats
stats.mannwhitneyu(y_train,y_test)
```

```
MannwhitneyuResult(statistic=4721116.0, pvalue=0.2054467977621907)
```

Рис.8. Распределение обучающей и тестовой выборок, тест.

Также в рамках обработки данных были заполнены недостающие(пустые) значения в соответствующих столбцах медианной, так как это были переменные с непрерывным распределнием, отличающимся от нормального.

Перед непосредственным проведением факторного анализа была произведена стандартизация данных.

3.4. Проведение факторного анализа

Факторный анализ был произведен для сокращения числа переменных, чтобы уменьшить количество вычислений и упростить последующие модели машинного обучения.

В ходе многократных попыток произведения анализа главных компомент, было решено остановиться на 18 факторах, которые объясняют почти 91 процент имеющихся данных.

```
print ('Explained variance by component: %s', pca.explained_variance_ratio_.sum())
```

```
Explained variance by component: %s 0.9059649984446704
```

Рис.9. Доля объясненной части исходных данных факторами

Так как исходных признаков было порядка 30, а факторов получилось 18, довольно сложно и затратно по времени исследовать, что именно в себя вобрали факторы, поэтому в таких ситуациях интепритация отходит на второй план.

3.5. Построение модели нейронной сети

Для построения модели нейронной сети использовалась библиотека PyTorch.

Архитектура нейронной сети представляет собой 5 слоев с 3 функциями активации ReLU и 1 функцией активации сигмоидой на первом слое, выходной слой - линейный. Более подробно конфигурацию нейронной сети можно увидеть на рисунке 10.

```
class WorkNet(torch.nn.Module):
    def __init__(self, n_hidden_neurons):
        super(WorkNet, self).__init__()

        self.fc1 = torch.nn.Linear(18, n_hidden_neurons)
        self.activ1 = torch.nn.Sigmoid()
        self.fc2 = torch.nn.Linear(n_hidden_neurons, n_hidden_neurons*2)
        self.activ2 = torch.nn.ReLU()
        self.fc3 = torch.nn.Linear(n_hidden_neurons*2, n_hidden_neurons*4)
        self.activ3 = torch.nn.ReLU()
        self.fc4 = torch.nn.Linear(n_hidden_neurons*4, 10)
        self.activ4 = torch.nn.ReLU()
        self.fc5 = torch.nn.Linear(10, 1)
    def forward(self, x):
        x = self.fc1(x)
        x = self.activ1(x)
        x = self.fc2(x)
        x = self.activ2(x)
        x = self.fc3(x)
        x = self.activ3(x)
        x = self.fc4(x)
        x = self.activ4(x)
        x = self.fc5(x)
        return x

    def inference(self, x):
        x = self.forward(x)
        return x
```

```
net = WorkNet(200)
```

Рис.10. Архитектура нейронной сети

В качестве функции потерь использовалась классическая для задач регрессии функция средней квадрат ошибки.

В качестве оптимизатора выступал Adam с классическим значениям параметра скорости обучения 10^{-3}

Обучение нейронной сети происходит с помощью эпох и мини-батчей.

Запускалось обучение алгоритма и анализировалось значение функции потерь на тренировочном датасете каждые 10 батчей, когда данное значение переставало меняться, что происходило примерно через 20 эпох, обучение останавливалась.

Чтобы посмотреть результаты метрики RMSE, её значения для тренировочного и тестового датасета записывались на каждой итерации.

Для предотвращения переобучения и определения оптимального значения метрики полезно смотреть на графики обучения.

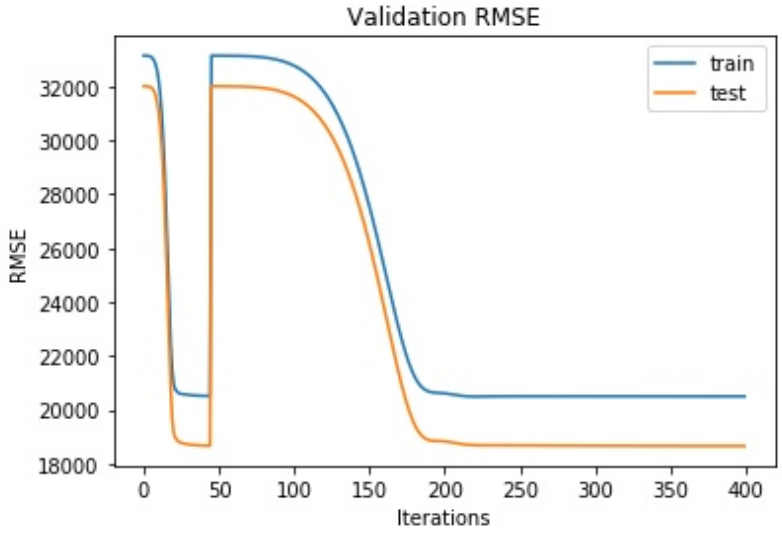

Рис.11. График обучения нейронной сети

На рисунке 11 заметно, что уже приблизительно на 40 итерации функция потерь и метрика RMSE достигают минимума на обоих датасетах, то же можно сказать и в окрестности 200 итерации.

3.6. Построение модели градиентного бустинга

Альтернативой нейронным сетям был выбран градиентный бустинг в модификации библиотеки xgboost как один из самых мощных методов при работе с табличными данными.

Несмотря на все свои преимущества, алгоритм имеет множество гиперпараметров и склонен к переобучению. Во избежание этого используется поиск по решетке гиперпараметров(GridSearch) с кросс-валидацией(в нашем случае с разбиением тренировочного датасета на 3)

```
n_trees = [x for x in range(1,101,10)]
max_depth = [x for x in range(1,10)]
learning_rate = [0.01,0.03]

param_grid = dict(n_estimators=n_trees, max_depth=max_depth, learning_rate=learning_rate)
model = xgb.XGBRegressor()
grid = GridSearchCV(estimator= model, param_grid=param_grid, scoring = 'neg_mean_squared_error',cv=3)
grid_result = grid.fit(Xtrain_transformed,y_train.numpy())
print("Best: %f using %s"% (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds= grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']

for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r"%(mean,stdev,param))
Best: -330517130.390112 using {'learning_rate': 0.03, 'max_depth': 3, 'n_estimators': 91}
```

Рис.12. Поиск по решетке и кросс-валидация

Лучшие параметры модели представлены на рисунке 12.

Также была предпринята попытка найти лучшее количество параметра nestimators(ко деревьев) в модели градиентного бустинга, а также построение графика обучения для предотвращения переобучения.

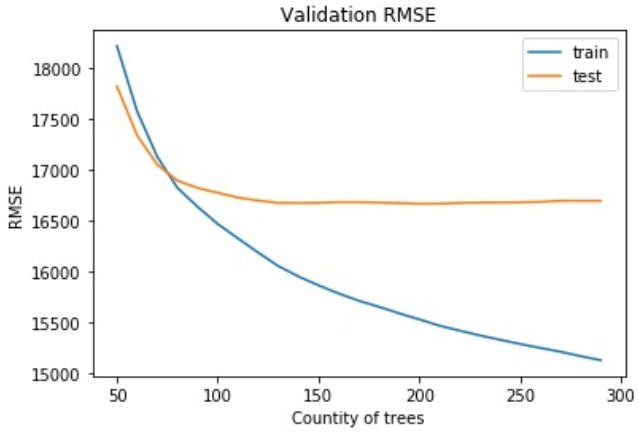

Рис.13. График обучения градиентного бустинга

На рисунке 13 заметно, что оптимальное количество деревьев находится в промежутке от 70 до 120, а затем метрика на тренировочном датасете существенно уменьшается, в то время как на тестовом немного возрастает, что говорит о переобучении.

Отметим, что показатель метрики RMSE у модели машинного обучения xgboost существенно ниже, чем у нейронных сетей.

4. Анализ результатов

Лучший результат нейронной сети оказался со значения RMSE 20509 для тренировочного датасета и 18670 для тестового.

В то время как градиентный бустинг при самых удачных гиперпараметрах, не процирующих переобучение, имеет лучший результат 16300 для тренировочного и 16724 для тестового

```
model = xgb.XGBRegressor(learning_rate = 0.03, max_depth = 3, n_estimators = 110, random_state = 42)

model.fit(Xtrain_transformed,y_train.numpy())
```

```
[15:08:52] WARNING: C:/Jenkins/workspace/xgboost-win64_release_0.90/src/objective/regression_obj.cu:
ecated in favor of reg:squarederror.

XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.03, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=None, n_estimators=110,
             n_jobs=1, nthread=None, objective='reg:linear', random_state=42,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
             silent=None, subsample=1, verbosity=1)
```

```
y_pred = model.predict(Xtest_transformed)

mean_squared_error(y_test,y_pred)**(1/2)
```

```
16723.538569945216
```

Рис.14. Результат метрики RMSE у лучшей модели

Но сама по себе метрика RMSE ничего не говорит, так как она привязана к конкретной задачи регрессии и к тому, в чем измеряется предсказываемый показатель.

В задачах же классификации, например, большинство метрик лежит от 0 до 1 и по ним можно однозначно сказать, насколько хороша модель или нет.

Для того, чтобы понять, насколько удачна построенная модель, рассмотрим распределение остатков.

Остатки представляют собой разницу между предсказанным моделью значением и реальным значением тестового набора.

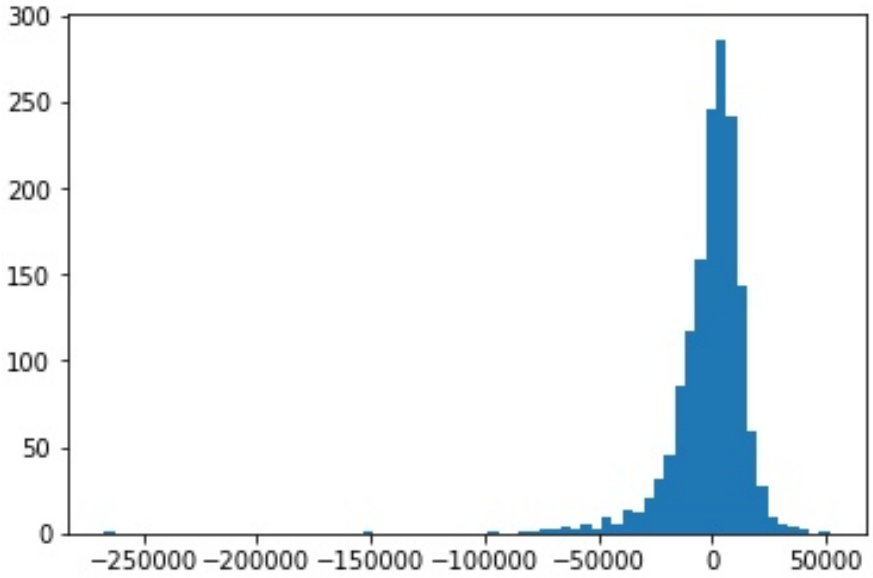

Рис.15. Гистограмма распределения остатков

На рисунке 15 заметно, что остатки имеют схожее распределение с нормальным, центр распределения находится в 0, что говорит о том, что модель довольно хорошо справляется с предсказанием истинного значения.

Хвосты распределения уходят плавно, без ассиметрий, что говорит о том, что модель не имеет склонностей(смещений) в предсказаниях.

Единственная слабость модели - это слабое предсказание некоторых высоких зарплат, о чем указывают выбросы в районе -100000 и меньше на гистограмме.

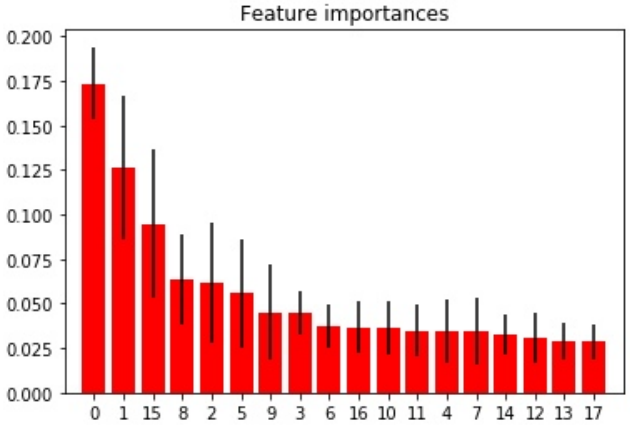

Рис.16. Важность факторов - признаков в модели xgboost

Также в качестве окончательного вывода о влиянии факторов на доход стоит обратить внимание на анализ featureimportance.

На рисунке 16 видно, что наибольшую значимость представляет собой 0 фактор, затем 1 и.т.д.

Чтобы понять, за что отвечают факторы, посмотрим на коэффицент корреляции между важными факторами и исходными признаками.(низкий коэффициент корреляции Пирсона не означает отстутсвие взаимосвязи в принципе)

Так, 0 фактор имеет высокую корреляцию с признаком "высшее образование"и высокую отрицательную корреляцию с признаком "среднее образование"

В то время как 5 фактор, например, имеет высокую корреляцию с признаком "количество рабочих часов в неделю".

Но как говорилось ранее, корреляции и взаимосвязь факторов и исходных признаков - вещь тонкая и не всегда поддается строгой интерпитации, а скорее интуиции.

```
for i in X_train.columns:
    k = pearsonr(X_train[i],df1[0])[0]
    if k > 0.7 or k<-0.7:
        print(k,i)
```

```
-0.7077947894220374 average
0.7457613685131221 high
```

Рис.17. Корреляция 0 фактора с исходными признаками

5. Заключение

В данной работе была использовалась среда Jupiter Notebook со встроенным языком Python. В нее загружались данные информационной базы исследования НИУ ВШЭ взятые за 2018 года для исследования человеческого капитала.

Из них были отобраны и проанализированы признаки, которые больше всего влияют на доход респондентов.

В дальнейшем эти признаки использовались для проведения факторного анализа, а также построения моделей машинного обучения, таких как нейронная сеть и градиентный бустинг.

Лучшей моделью оказалась модель градиентного бустинга. Исходя из значений метрики RMSE и анализа распределения остатков, можно отметить, что модели довольно хорошо удается предсказывать возможный доход человека по имеющимся факторам.

Самыми важными признаками, влияющими на доход респондента, оказались: наличие высшего образование, количество часов работы в неделю и месяц, пол, знание иностранного языка, место рождения(городская среда или сельская).

6. Список использованных источников

1. Николенко С. И., Кадурин А.Б. «Глубокое обучение - Погружение в мир нейронных сетей.», 2018. Сетевое электронное издание учебного пособия. 300 страниц, формат PDF.

2. PyTorch Introduction https://pytorch.org/