

Правительство Российской Федерации Федеральное государственное автономное  
образовательное учреждение высшего образования  
Национальный исследовательский университет  
«Высшая школа экономики»

Образовательная программа «Прикладная математика»

Отчет

по проектной работе

**Прогнозирование зарплатных ожиданий соискателей с  
помощью методов машинного обучения**

Карнаушенко Михаил  
Цыплаков Александр

Москва 2022

# Отчёт о проделанной работе по теме "Прогнозирование зарплатных ожиданий соискателей с помощью методов машинного обучения"

Карнаушенко Михаил  
Цыплаков Александр

2 июня 2024 г.

## 1 Введение

### 1.1 Актуальность

Интеллектуальная система, представленная в данном проекте, значительно упрощает работу по поиску сотрудников для различных компаний, ровно как и наоборот, помогает кандидатам при выборе вакансий. Такая система основывает свой поиск на независимой оценке способностей, а также делает упор на умение работать с большим количеством данных.

### 1.2 Практическое применение

Данный проект носит как теоретический, с точки зрения получаемых знаний и статистического анализа, так и практический характер, буквально помогающий получать данные о зарплатных ожиданиях.

## 2 Цель

Научиться применять знания о машинном обучении на практике.  
Научиться собирать и анализировать данные для интеллектуальных моделей.  
Научиться прогнозировать зарплатные ожидания соискателей с помощью машинного обучения.

## 3 Задачи

- Сбор данных.
- Обработка данных.
- Обучение модели.
- Оценка качества работы и сравнение результатов.

## 4 Ход работы над проектом

### 4.1 Сбор данных

Мы решили собирать данные с сайта hh.ru[?] - популярного в России сервиса для поиска работы и сотрудников. Рассматривались и другие варианты, но мы посчитали, что этот сайт наиболее удобен для сбора информации. Поскольку данные анализировать целесообразно по одной категории, то мы выбрали одну из популярных и актуальных сфер, такую как 'IT-специалист'. Для этого в графе поиска на сайте выберем интересующую нас профессию и выставим фильтры так, чтобы как можно больше категорий отображалось в резюме.

По-скольку получать данные вручную было бы крайне долго, то нами был выбран вариант парсинга данных с сайта. Парсинг - это автоматизированный сбор информации с веб-ресурса. В нашем случае, все программы для этих целей мы писали на языке программирования Python, поскольку он обладает большим количеством библиотек, подходящих для данной задачи[?]. Основные библиотеки, используемые нами:

- requests
- selenium
- BeautifulSoup
- fake-useragent
- lxml
- NumPy
- sklearn
- matplotlib
- Pandas

Каждая библиотека отвечает за определенный этап парсинга. Библиотека requests отправляет запрос на сайт, как если бы это делал человек. Но, как правило, сайты не любят, когда на них отправляется много запросов за короткий промежуток времени. Именно поэтому мы подключили еще одну библиотеку fake-useragent, которая генерирует фейковые user-agent (буквально маски, под которыми мы заходим в интернет). Тем самым сайт больше не блокирует наш поток запросов. Дальнейшая проблема, с которой мы столкнулись - динамический сайт. То есть, при открытии страницы, ее содержимое появляется и становится видимым для "считывания" HTML-кодом только после прокрутки пользователем страницы вниз. Для этой цели мы подключаем библиотеку selenium, способную работать с пользовательским интерфейсом. Порядок выполнения таков:

Запрос отправляется на сайт -> selenium прокручивает страницу до ее конца -> HTML-код сохраняется к нам в архив

hh.ru предлагает услуги предоставления всей своей базы данных платно, поэтому мы ограничились 5000 резюме для общего доступа. Структура сайта такова, что происходит загрузка страницы по 100 резюме в кратком виде, таким образом нам было доступно 50 страниц с 100 резюме на странице. Каждое резюме не имело достаточной информации, но имело ссылку на полное резюме. Поэтому дальше мы подключили библиотеку BeautifulSoup, которая открывала по очереди все 5000 резюме и сохраняла код страницы. Дальше с помощью парсера lxml и BeautifulSoup мы собирали данные с каждого резюме.

## 4.2 Обработка данных

Изначально нам показалось, что данных для обучения модели будет недостаточно, так как для каждого из 5000 объектов находилось не больше 10 признаков. С этой проблемой мы начали искать готовые данные по нашей теме, но нашли лишь базы с меньшим количеством признаков, чем у нас. Было принято решение собирать все имеющиеся данные, указанные в резюме.

- wage - заработная плата, то есть тот показатель, который мы хотим получить как ответ на нашу задачу.
- age - возраст соискателя.
- exp - опыт работы соискателя в годах.
- sex - пол соискателя
- bachelor - высшее образование
- special education - колледж

- courses - факт прохождения курсов повышения квалификации
- English - знание языка
- driver license - наличие водительских прав
- private car - наличие собственной машины
- business trip - готовность к командировкам
- relocate - готовность к переезду
- travel time - требование к времени на дорогу
- skills - навыки соискателя

Среди полученных данных присутствуют как численные значения (то есть интервальные признаки), так и категориальные признаки, отвечающие за принадлежность объекта к тому или иному классу. То есть, например, признак `exp` - интервальный, а `sex` - категориальный. Так как компьютер способен обрабатывать только числа, то был применен метод One-Hot-Encoding[?], который разбивает признак на внутренние подпризнаки, формируя матрицы. То есть признак `English` будет выглядеть как вектор:

$$\text{English} = \begin{pmatrix} A1 \\ A2 \\ B1 \\ B2 \\ C1 \\ C2 \end{pmatrix}$$

Здесь каждая строка в вектор-столбце отвечает за определенный уровень владения английским языком и принимает значение 0 или 1. Также, после получения всех данных, мы можем их нор-

мировать, то есть привести к какому-то диапазону значений. Это необходимо, чтобы значения, имеющие большой показатель отклонения от стандартной величины не портили точность прогнозов. Такое действие не обязательно для всех моделей, но для некоторых важно. Точность можно оценивать, используя метрики.

#### 4.2.1 Обучение Модели

Так как мы хотим прогнозировать зарплатные ожидания, то нами была выбрана регрессионная модель обучения, которая дает на выходе численный ответ - зарплату, на которую мог бы рассчитывать соискатель. Для этих целей мы выбрали 2 модели, которые будем обучать и сравнивать результаты работы:

- Линейная регрессия
- Случайный лес

Подробнее про принцип работы всех двух моделей будет написано далее. При первой попытке написания модели линейной регрессии мы столкнулись с серьезной проблемой

Она заключалась в том, что построение линейной регрессии больше подходит для интервальных переменных, а не категориальных, то есть на графике рассеивания зарплаты от опыта работы(или возраста), располагаются точки (люди), а далее проводится прямая, которая будет проходить через центр этого облака. Таких переменных было всего две: возраст и стаж работы, но, как оказалось далее, корреляция между зарплатой и одним из этих признаков слишком мала для построения точной модели, к тому же возникла **мультиколлинеарность** между этими двумя предикторами, то есть независимые переменные очень сильно коррелировали между собой.

РРР,,РҘРёР«Рэ СҘРэСГРёР,,Рө 2022-05-23 Рҗ 21.07.28.jpg

Корреляция между зарплатой, стажем работы и возрастом.

График рассеивания между зарплатой и опытом работы.

Как раз из-за мультиколлинеарности и возникла проблема построения модели по этим данным. Именно по этому было принято решение построить модель по категориальным данным, используя **Ridge** регрессию, о которой будет написано далее. После построения модели по такому принципу, метрика  $R^2$ , которая также будет описана позже, была равна 0.12, что, естественно, не удовлетворяет нашим требованиям. Поэтому мы вернулись к изначальному датасету и стали вручную анализировать данные и пришли к выводу, что люди по большей части указывали зарплату вне зависимости от собственных навыков, то есть, к примеру, у человека слишком завышенная зарплата, а особые навыки отсутствуют, и наоборот. В следствие этого нам пришлось отказаться от собранного датасета.

#### 4.2.2 Сбор новых данных и их обработка

Так как наши данные не прошли проверку на пригодность, мы решили собрать новые. Повторив уже существующую схему парсинга, мы собрали еще несколько датасетов, но уже для других отраслей, и с других сайтов[?]. Но эти датасеты вновь не превысили минимальных ожиданий. Именно поэтому, за неимением других вариантов, мы решили взять готовый датасет. Наш выбор пал на данные с сайта предоставляющего большое количество уже собранных данных[?].

Поскольку тема проекта - прогнозирование зарплатных ожиданий, то мы решили взять таблицу с информацией о 14447 работниках институтов, для каждого из которых указаны такие данные: имя, должность, зарплата, отдел и название колледжа.

output\_density\_salaries.png

График зависимости зарплаты от количества человек

Очевидно, что зарплата от имени никак не зависит, поэтому мы смело можем удалить этот столбец из датасета, не боясь, что это как-то повлияет на нашу точность.

С помощью One-Hot-Encoding[?], встроенного в sklearn кодируем все категориальные признаки, которые у нас есть. Причем также считается количество вхождений закодированного слова (то есть его частота появлений, которая может быть показателем для возможности устранения из датасета). После кодировки получаем такое разбиение: столбец с должностью разбивается на 147 закодированных колонок, каждая колонка является определенной должностью и принимает значение либо 1, либо 0; столбец с отделом в колледже разбивается на 109 закодированных, по такому же принципу, колонок; столбец с колледжем разбивается на 12 закодированных колонок соответственно.

#### 4.3 Обучение модели

Две модели, которые мы будем использовать отличаются по своей структуре.

- Линейная регрессия (Ridge)

Так как мы уже выяснили, что предыдущий датасет не подошел нам, то мы решили использовать линейную регрессию с L2 регуляризатором, или линейную регрессию типа ridge[?]. Ее особенность заключается в том, что она сглаживает неравенство весов коэффициентов у признаков. Формульно ее можно задать так:

$$(y - wX)^2 + \lambda \sum w_i^2 \rightarrow \min$$

В отличие от обычной линейной регрессии без регуляризации:

$$(y - wX)^2 \rightarrow \min$$

Здесь  $y$ ,  $X$  задают целевую переменную и матрицу признаков, а  $w$  - их веса, то есть коэффициенты. А в общем уравнение отвечает методу наименьших квадратов (МНК).

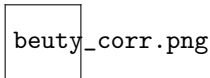
- Случайный лес

Случайный лес[?]- предсказательная модель, строящая множество деревьев решений и выдающая средний результат. Дерево решений - дерево, которое в каждом своем узле содержит условное выражение, делящее выборку. Такая модель очень хорошо дает прогнозы, но плохо расставляет закономерности. То есть нет такого явного набора коэффициентов, как в случае с линейной регрессией.

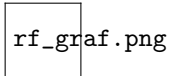
Теперь, когда мы окончательно определились с моделями, мы можем начать процесс обучения. Для этого нам понадобится по большей части библиотека `sklearn` в `python`:

```
sklearn.preprocessing: OneHotEncoder
sklearn.compose: ColumnTransformer
sklearn.pipeline: Pipeline
sklearn.model_selection: KFold
sklearn.linear_model: Ridge
sklearn.ensemble: RandomForestRegressor
```

Для обучения данные делятся на тренировочную и тестовую выборку. Соответственно, модель учится предсказывать на тренировочных данных, и делает предсказания на тестовых. Но возникает проблема - так как данные могут различаться и быть упорядочены, то модель, учившаяся на одних значениях, будет плохо работать на других. Для этого существует кросс-валидация, которая делит их на несколько частей, поочередно выбирая их как тестовые и тренировочные. Предварительно мы специально перемешиваем данные, чтобы избежать какой-либо упорядоченности в датасете. После этого предсказания усредняются, что позволяет нам сравнивать результаты работы разных моделей. Мы использовали деление на 3, 5, 10 и 20 частей. Чем больше было количество разбиений - тем выше становилась точность, но повышать количество разбиений до слишком высоких значений уже сложнее - не хватает вычислительных мощностей. Вот несколько примеров работы моделей:

A small square placeholder image with the text "beuty\_corr.png" inside.

Здесь по горизонтали указаны люди, а по вертикали - их зарплаты. Голубым цветом отмечены истинные тестовые данные, красным - прогнозы линейной регрессии.

A small square placeholder image with the text "rf\_graf.png" inside.

Здесь по горизонтали указаны люди, а по вертикали - их зарплаты. Зеленым цветом отмечены истинные тестовые данные, красным - прогнозы случайного леса.

Мы написали функцию, которая выводит тонкость модели, и запустили ее для всех наших моделей. Получив результат, мы можем переходить к выводам.

#### 4.3.1 Метрики

Мы использовали три метрики[?]:

- MAPE
- RMSE
- $R^2$

*MAPE* - Mean absolute percentage error regression loss[?], средняя абсолютная процентная потеря регрессии. Показывает на сколько велики ошибки в сравнении со значениями ряда. Вычисляется по формуле:

$$MAPE = \frac{1}{n} \sum \frac{|Y_{test} - Y_{pred}|}{Y_{test}} \quad (1)$$

Здесь  $Y_{test}$  - тестовые значения из выборки,  
 $Y_{pred}$  - значения предсказаний,  
 $n$  - количество значений в выборке.

$RMSE$  - Root Mean Squared Error[?], то есть среднеквадратичная ошибка. Показывает то, насколько в среднем отклоняется величина от среднего значения. Вычисляется по формуле:

$$RMSE = \sqrt{\frac{\sum(Y_{test} - Y_{pred})^2}{N}} \quad (2)$$

Здесь  $Y_{test}$  - тестовые значения из выборки,

$Y_{pred}$  - значения предсказаний,

$N$  - количество значений в выборке. Чем меньше это число, тем больше точность нашей модели.

$R^2$  - обозначает коэффициент детерминированности (R-Squared[?]) и выражается формулой:

$$R^2 = 1 - \frac{\sum(Y_{test} - Y_{pred})^2}{(\sum Y_{test} - Y_{avg})^2} \quad (3)$$

Здесь  $Y_{test}$  - тестовые значения из выборки,

$Y_{pred}$  - значения предсказаний,

$Y_{avg}$  - среднее значение из тестовой выборки. Чем больше значение  $R^2$  к числу 1, тем больше точность нашей модели.

Все формулы для подсчета метрик уже есть в качестве встроенных функций библиотеки sklearn.

## 4.4 Результаты

Для всех моделей приведем полученные результаты метрик.

Модель	MAPE	RMSE	$R^2$
Линейная регрессия	0.420	$133.00 \pm 1.40$	$0.638 \pm 0.012$
Случайный лес	0.320	$123.97 \pm 1.57$	$0.626 \pm 0.013$

## 5 Выводы

Анализируя полученные результаты, мы можем прийти к выводу, что стандартное отклонение метрики RMSE для линейной регрессии и случайного леса очень близки, в то время как абсолютное значение этой метрики для двух моделей отличается примерно на 7%. Значение же метрики MAPE меньше для модели случайного леса, то есть он выдает более точный результат. Исходя из этого можно сделать вывод о том, что предсказания случайного леса подходят для нашей задачи наилучшим образом. Дело в том, что случайный лес более аккуратно учитывает категориальные признаки. Точность ( $R^2$ ) при различных значениях разбиений для кросс-валидации достигла значения, близкого к 64%. Таким образом мы научились собирать данные для интеллектуальных моделей, научились их правильно фильтровать и строить на их основании модели, принцип которых мы понимаем.

## 6 Распределение обязанностей группового проекта

- Первая сборка

Михаил	Александр
Парсинг данных с сайта hh.ru	Парсинг данных с сайта работа.ru
Обучение модели случайного леса	Обучение модели линейной регрессии
Расчёт метрики RMSE, $R^2$ и MAPE для случайного леса	Расчёт метрики RMSE, $R^2$ и MAPE для линейной регрессии

- Вторая сборка

Михаил	Александр
Обучение модели случайного леса	Обучение модели линейной регрессии
Расчёт метрики RMSE, $R^2$ и MAPE для случайного леса	Расчёт метрики RMSE, $R^2$ и MAPE для линейной регрессии

## Список литературы