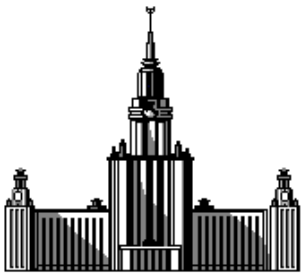
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет Вычислительной Математики и Кибернетики

Кафедра Исследования Операций



“Data Science в экономике: корреляционный и регрессионный анализ и прогнозирование экономических показателей”

Курсовая работа
студента 312 группы
Денисова Никиты Сергеевича

Научный руководитель:
Оленёв Н.Н.

Москва – 2020

Оглавление:

Введение.....……3

Постановка задачи……...3

Основные этапы решения поставленной задачи……...3

Подготовка данных……..4

Корреляционный анализ……5

Прогнозирование курса доллара по совокупности признаков……6

- **XGBoost Regressor……7**
- **SKLearn Linear Regression……..7**
- **SKLearn Ridge/Lasso Regression……8**
- **Гребневая регрессия……8**
- **Лассо регрессия……8**
- **SKLearn KNeighboorRegressor……...9**
- **SKLearn RandomForestRegressor……..9**
- **Итога этапа прогнозирования на основе набора признаков……..10**

Прогнозирование каждого из признаков на основе их предыдущих значений……..10

- **Прогнозирование признаков с помощью ARIMA……10**
- **Прогнозирование признаков с помощью SKLearn……..12**

Заключение и выводы……...13

Источники……...14

Введение

Data Science — обширная область, покрывающая анализ данных, машинное обучение, нейронные сети, проектирование данных. В данной работе я провожу анализ связей между различными экономическими показателями, их влияния друг на друга, закономерностей в них, а также строю несколько моделей, которые предсказывают необходимые данные. Датасетами (то есть наборами данных) являются цены на нефть, значения различных биржевых индексов и курс рубль-доллар, который в конечном итоге будем прогнозировать, основываясь на первых двух типах данных. В качестве инструмента анализа и прогнозирования используется язык Python и необходимые библиотеки для машинного обучения, статистики, построения различных графиков.

Постановка задачи

Спрогнозировать курс рубль-доллар на основе набора значений признаков: цены на баррель нефти Brent и значений различных биржевых индексов. Реализуется средствами Python и различных библиотек.

Основные этапы решения поставленной задачи

1. сбор и подготовка данных для обучения различных моделей

2. анализ корреляции различных признаков для отбора только значимых. Детальное изучение зависимости между российскими биржевыми индексами/ценами на нефть и курсом рубль доллар

3. нахождение функции $f(x_1, \dots, x_n) = Y$ для прогнозирования зависимой величины Y (стоимость доллара в рублях) на основе имеющихся признаков. На данном этапе предполагается изучение возможностей регрессии различных библиотек и обучение соответствующих моделей на имеющихся статистических данных. Далее планируется отбор наиболее подходящих моделей на основе коэффициента детерминации (метрики R^2), который показывает, насколько прогнозируемые величины близки к реальным. Вычисляется он по формуле:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_t - \widetilde{y_t})^2}{\sum_{i=1}^{n}(y_t - \overline{y_t})^2}$$

   Здесь $\widetilde{y_t}$ – спрогнозированные значения ответов, а $\overline{y_t}$ – среднее по исходной выборке ответов. Коэффициент R^2 лежит в диапазоне [0 , 1], и чем он ближе к 1, тем точнее прогнозы модели. Таким образом, имея значения признаков на текущий день, можем спрогнозировать курс рубль-доллар также на текущий день.

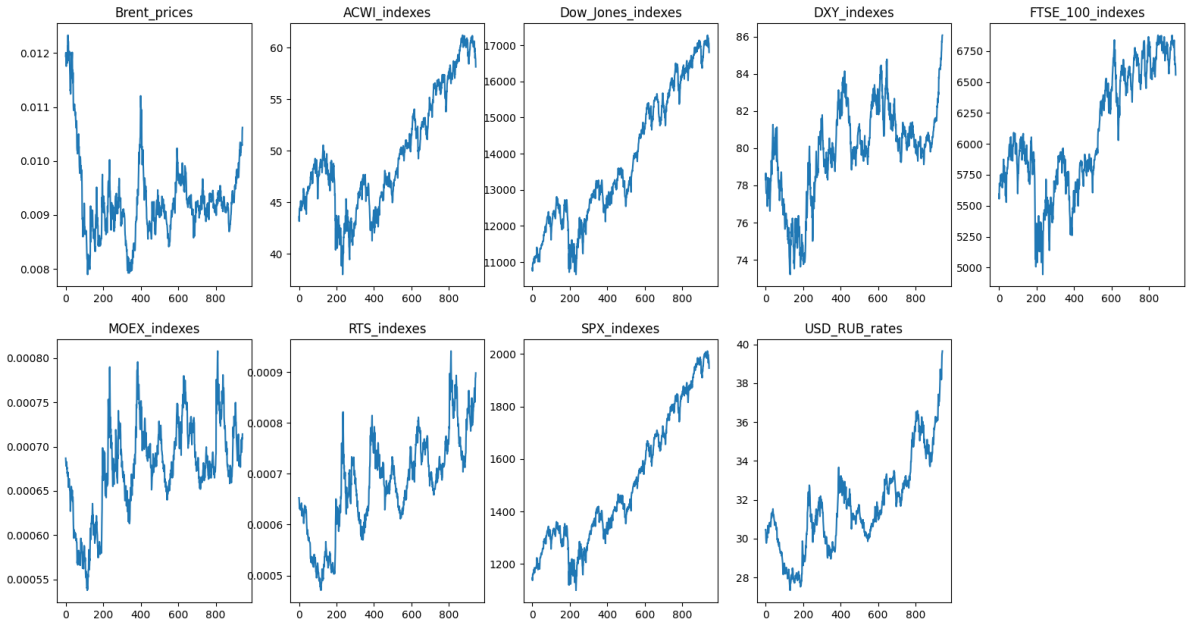
4. Чтобы спрогнозировать курс рубль-доллар на следующий день, необходимо спрогнозировать все признаки на следующий день. Для этого используются 2 подхода: статистический и машинное обучение. В статистическом походе во временном ряде находятся тренд, сезонность и шум, а также делается его анализ на стационарность. Если она отсутствует, то производится дифференцирование временного ряда до тех пор, пока он не станет стационарным. Когда временной ряд удовлетворяет всем условиям, применяется модель прогнозирования ARIMA (autoregressive integrated moving average). При машинном же обучении используется

все те же модели регрессии, которые были задействованы в пункте 3. Отбор оптимальных моделей происходит также посредством коэффициента детерминации.

Изучим более детально, что необходимо делать на каждом из этапов.

Подготовка данных

В самом начале необходимо собрать все потенциально необходимые данные. В общем случае некоторые из них впоследствии могут оказаться лишними (почему – станет ясно на втором этапе), но в нашем случае мы будем руководствоваться некоторыми экономическими соображениями. Экономика России – сырьевая, поэтому курс рубль-доллар сильно зависит от цен на нефть, поэтому возьмем данные по стоимости барреля нефти марки Brent в период с 2000 по 2020 год. Данные скачиваются в формате csv с сайта investing.com, поэтому с одной из проблем – откуда брать датасеты для нашей задачи – мы не столкнулись, потому что данный сайт предоставляет богатый функционал для отслеживания всех биржевых показателей. Другой показатель, оказывающий достаточное влияние на курс, – это национальный индекс. В нашем случае мы будем рассматривать два главных индекса РФ – РТС и ММВБ (индекс Мосбиржи). Так как мы рассматриваем пару рубль доллар, то не лишним будет взять в рассмотрение и индексы США – Dow Jones и S&P 500. Также полезно будет включить в нашу модель индекс доллара – эдакий критерий для оценки состояния Американского доллара. Как и другие активы на бирже, он тоже торгуется. Дополнительными индексами в нашей модели будут: британский FTSE100, в который входит British Petroleum, и всемирный ACWI – all country world index, который отражает состояние мирового фондового рынка. Таким образом, для прогнозируемой величины курс рубль-доллар мы отобрали 8 признаков: стоимость нефти и 7 различных индексов. Так выглядят необходимые показатели в период с 2010 по 2014 год (стоит обратить внимание на диапазон значений стоимости нефти и двух индексов: ММВБ и РТС):



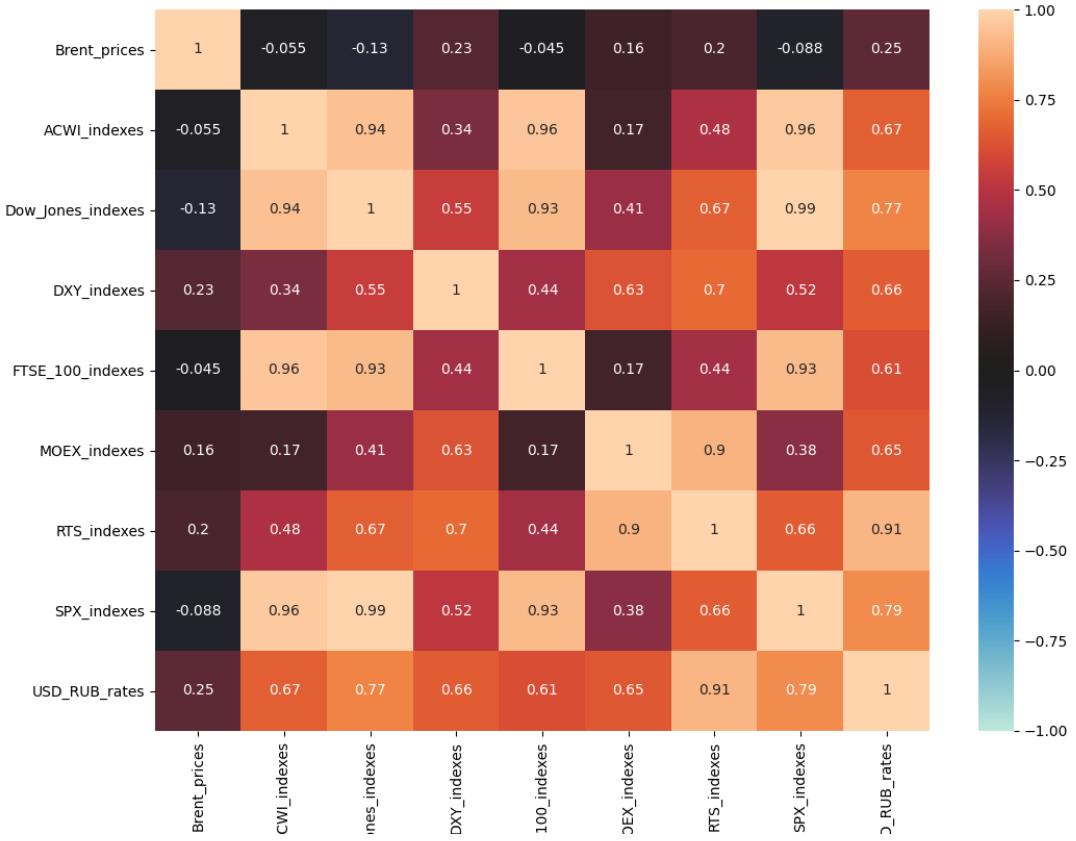
Каждый из показателей находится в своем csv файле, в котором есть цена за день, цена открытия и закрытия, процент изменения и т.д. Нас интересует лишь дневная цена, поэтому сделаем пересечение всех таблиц по столбцу даты и получим равномерные временные ряды для каждого из признаков (под равномерностью подразумевается то, что для каждой даты имеются значения всех признаков).

Корреляционный анализ

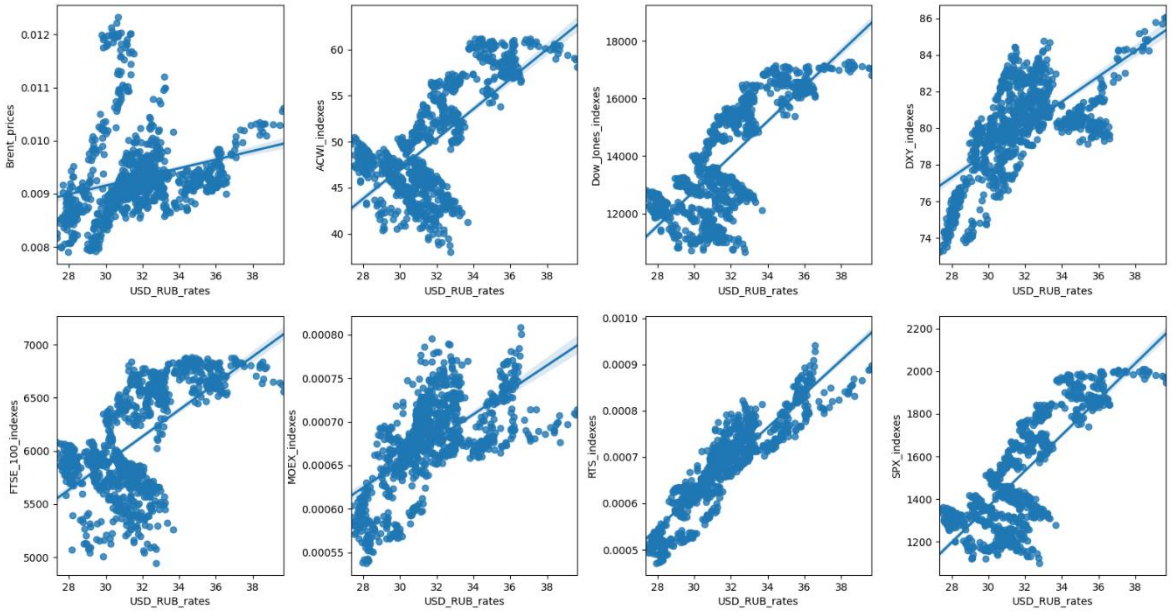
На этом этапе необходимо понять, насколько связаны признаки и прогнозируемая величина. Для начала отмечу, что всё обучение и подбор моделей производится на данных за 2010 — 2014 года, так как в тот период курс рубль-доллар и стоимость нефти ведут себя достаточно стабильно. В качестве метрики корреляции будем использовать коэффициент корреляции Пирсона, вычисляемый по формуле:

$$r_{XY} = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2 (Y_i - \overline{Y})^2}}$$

где $\overline{X}, \overline{Y}$ – среднее каждой из выборок. Он лежит на отрезке [-1, 1], положительные значения коэффициента говорят о прямой корреляции, отрицательные – об обратной, а модуль коэффициента говорит о “силе” корреляции. На основе данного показателя мы можем судить о значимости каждого признака при прогнозировании целевой величины. На коррелограмме ниже нас интересует последняя строка:



Можем видеть, что со всеми величинами, кроме цены на нефть, курс рубль-доллар достаточно коррелирует. Но так как экономика России – сырьевая, то не исключаем нефть из признаков, так как корреляция зависит и от временного периода, то есть с 2014 по 2018 зависимость может быть совершенно другой, что объясняется подвижностью экономики. Как было отмечено в разделе про подготовку данных, три из всех признаков принимают непривычные значения. Этому есть причина – национальные биржевые индексы показывают силу валюты, и чем они выше, тем, применительно к России, стоимость доллара должна снижаться или, как минимум, не расти. Таким образом, для всех значений *x(t)* каждого из трех признаков: стоимости нефти и двух индексов, которые являются временными рядами, была взята обратная величина – *1/x(t)*. Таким образом корреляции со всеми признаками (и исходными, и модифицированными) – прямые, и это позволяет нам обучать на этих данных равные модели. Рассмотрим также следующую картинку, где координата каждой точки – *(значение признака в момент t, стоимость доллара в рублях в момент времени t)*:



Чем лучше распределены точки вдоль диагонали y = x, тем сильнее корреляция. Можно видеть, что хорошая корреляция имеется лишь с обратным значением индекса РТС. Это отчасти говорит о непредсказуемости в экономике, о наличии спекуляций и необъяснимых движений. Но в целом картина удовлетворительная, поэтому можно переходить к следующему этапу – прогнозированию доллара по совокупности признаков.

Прогнозирование курса доллара по совокупности признаков

На этом и следующем этапе главным инструментом будет Python и различные библиотеки: Scikit-learn, XGBoost, Statsmodels и др. Для обучения моделей исходный набор данных нужно разбить на матрицу значений признаков X и вектор столбец y, где каждая строка X – значения признаков в момент времени *t*, а *y(t)* – отвечающая им стоимость доллара в рублях. Данная задачи является множественной регрессией, и для её решения будем использовать различные модели: линейную регрессию, гребневую и лассо регрессии, метод k-ближайших соседей, случайный лес, экстремальный градиентный бустинг. Рассмотрим подробно каждую из них.

XGBoost Regressor

В библиотеке XGBoost, в том числе и классе XGBRegressor (который, очевидно, используется для регрессии), реализуется метод экстремального градиентного бустинга. Бустинг — это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов. Бустинг представляет собой жадный алгоритм построения композиции алгоритмов. Суть обучения модели – минимизация функции потерь. В случае задачи регрессии можно использовать средний квадрат ошибки:

$$Loss = MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_i^p\right)^2$$

где y_i – реальное значение, а y_i^p – предсказанное. Суть градиентного бустинга – использование градиентного спуска и обновление предсказаний для минимизации потерь по формуле:

$$y_i^p = y_i^p + \alpha * \frac{\delta \sum_{i=1}^{n}\left(y_i - y_i^p\right)^2}{\delta y_i^p}$$ (δ – знак частной производной)

$$\Downarrow$$

$$y_i^p = y_i^p - \alpha * 2 * \sum_{i=1}^{n}\left(y_i - y_i^p\right)$$

где α – скорость обучения, то насколько сильно корректируются предсказания. По сути каждое $y_i^p = F(x_1, \dots, x_n), y_i^p \in R$, поэтому, дифференцируя функцию потерь по каждому x_i, получаем систему, которая решается итерационно до достижения заранее заданной точности. Главная проблема, которую нужно избегать – это переобучение, поэтому в XGBRegressor заложены параметры для L2 и L1 регуляризации. Таким образом, данная библиотека представляет собой улучшение изначального алгоритма градиентного бустинга, которая также включает себя различные аппаратные оптимизации (параллельное построение деревьев, оптимальное использование ресурсов и т.д.). Класс XGBRegressor имеет широкий набор параметров: количество деревьев, их максимальная глубина, скорость обучения, вид алгоритма, который используется в композиции, параметры регуляризации и др. Сложно заранее установить параметры так, чтобы обученная модель давала наилучшую точность прогноза, поэтому для поиска оптимального алгоритма используется перебор по сетке гиперпараметров в совокупности с кросс-валидацией, чтобы в итоге, на основании некоторой метрики, отобрать ту модель, у которой эта метрика будет наилучшей. Здесь и далее не считаю нужным приводить отрывки кода, потому что программа и данные будут в приложениях, а последовательность действий в коде достаточно понятна.

SKLearn Linear Regression

Одной из самых простых и легко интерпретируемых моделей машинного обучения является линейная регрессия. Задается она нехитрым уравнением:

$$Xw = y$$

где *X* – матрица значений признаков, *w* – вектор-столбец весов, в котором w_i соответствует *i-ому* признаку, а *y* – целевое, прогнозируемое значение, вектор-столбец ответов. В общем

случаем w и y могут быть не векторами, а матрицами. В нашем же случае $X \in R^{n*m}, w \in R^m, y \in R^n$. Функция потерь, которую будем минимизировать для нахождения оптимальных весовых коэффициентов, имеет вид:

$$Loss = MSE = Q = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m} x_{ij}\, w_j - y_i\right)^2$$

Беря частные производные функции потерь по весовым коэффициентам и приравнивая их к нулю (необходимое условие минимума функционала):

$$\frac{\delta Q}{\delta w_i} = 0$$

получаем СЛАУ. Далее, используя один из итерационных методов, получаем приближенное решение с заданной наперед точностью. Точное же решение получить можно крайне редко.

SKLearn Ridge/Lasso Regression

Дабы избежать переобучения, гребневая и лассо регрессия накладывают ограничения на функцию потерь, которую мы минимизируем.

Гребневая регрессия – вид регрессии с дополнительными ограничениями. Если в матрице X какие-либо признаки коррелируют между собой, то возникает мультиколлинеарность, что влечет за собой неустойчивость оценки коэффициентов регрессии. Поэтому вводится модифицированный функционал ошибки:

$$Q = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m} x_{ij}\, w_j - y_i\right)^2 + \theta\sum_{i=1}^{n} w_i^2$$

где θ – параметр *L2* регуляризации. Гребневая регрессия не производит отбор признаков, а лишь уменьшает значения коэффициентов для некоторых из них.

Лассо регрессия – вид регрессии с понижением размерности. Также, как и гребневая регресия, лассо регрессия применяется в случае корреляции признаков, но в ней используется другой вид штрафа. Функционал ошибки выглядит следующим образом:

$$Q = \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{m} x_{ij}\, w_j - y_i\right)^2 + \lambda\sqrt{\sum_{i=1}^{n} w_i^2}$$

где λ – параметр *L1* регуляризации. В данном виде регрессии уже производится отбор признаков, то есть некоторые w_i будут равны нулю.

У каждого из этих двух видов регрессии есть свои преимущества, поэтому выбор того или иного алгоритма зависит от поставленной задачи. Конкретно в моем случае гребневая регрессия дала более высокий коэффициент детерминации, нежели лассо регрессия, несмотря на наличие корреляции между некоторыми признаками (см. таблицу корреляции).

Как и в случае с XGBRegressor, для Ridge и Lasso регрессий производится подбор оптимальных параметров путем поиска по сетке гипермараметров.

SKLearn KNeighboorsRegressor

Следующий рассматриваемый алгоритм – метод k-ближайших соседей. Достаточно простой метрический регрессор (а также классификатор), которой присваивает прогнозируемой величине значение, равное среднему по k-ближайшим соседям. Так же, как и в предыдущих моделях, имеется матрица признаков X и вектор ответов y. Предполагается, что задана функция расстояния между двумя объектами из X: $\rho(x, x'), x, x' \in X$, являющаяся адекватной моделью сходства. Чем больше значение данной функции, тем дальше друг от друга расположены объекты. Для произвольного объекта u расположим объекты обучающей выборки x_i в порядке возрастания их расстояния до u: $\rho(u, x_{1;u}) \leq \rho(u, x_{2;u}) \leq \cdots \leq \rho(u, x_{n;u})$, где $x_{i;u}$ – i-ый сосед для объекта u. Значение, соответствующее объекту u, вычисляется по формуле:

$$a(u) = argmax \sum_{i=1}^{n} [y(x_{i;u}) = y] w(i, u)$$

где максимум берется по всем $y \in Y$, а $w(i, u)$ – весовая функция, то есть коэффициент, с которым i-й объект обучающей выборки должен войти в прогнозируемое значение. Различают несколько весовых функций:

- Метод ближайшего соседа: $w(i, u) = [i = 1]$
- Метод k-ближайших соседей: $w(i, u) = [i \leq k]$
- Метод k экспоненциально взвешенных ближайших соседей: $w(i, u) = [i \leq k] q^i$
- Метод парзеновского окна и метод потенциальных функций

Первую из них не рекомендуется использовать, так как она крайне неустойчива к выбросам, а в третьем предполагается, что $q < 1$. Оптимальный параметр k подбирается путем кросс-валидации. Считается полезным исключить из выборки те объекты, вокруг которых плотно (в смысле выбранной метрики) расположены другие объекты, так как это ведет не только к уменьшению количества хранимой информации, но и к отсеву несущественных объектов. Также для улучшения алгоритма нелишним считается нормировка значений признаков, иначе большие значения некоторых из них будут перекрывать незначительные (но важные для нас) отклонения в других. Еще одна из возможных проблем – высокая размерность признакового пространства, так как по закону больших чисел суммы большого числа небольших отклонений будут давать похожие значения, что ведет к практически произвольному выбору нежных соседей. Класс KNeighboorsRegressor позволяет учесть все эти аспекты и, после выбора оптимальных параметров из сетки, показывает неплохой результат.

SKLearn RandomForestRegressor

Рассматриваемый алгоритм относится к ансамблевым методам, был придуман после создания CART – classification and regression trees. Как можно видеть из его названия, строится композиция решающих деревьев, на основе которой делается прогноз. В результате многочисленных экспериментов было установлено, что точность гораздо выше у ансамблей, нежели у отдельных деревьев. Каждое дерево из ансамбля – бинарное, решающее, в каждом узле которого (кроме листьев) находится целевая функция. На каждом этапе делается случайная выборка объектов для обучения и тестирование, а прогноз делается путем взятия среднего по всем предсказаниям. Такой алгоритм называется бэггинг (англ. *bagging*). Каждое регрессионное решающее дерево строится по следующему алгоритму:

- Выбирается целевая функция: энтропийный индекс, индекс Джини или др.
- На основе некоторого показателя этой функции (математическое ожидание, дисперсия и т.д.) делается переход к следующей вершине
- Достижение критерия остановки, то есть попадание в лист. Таким критерием может быть неразделимость по всем признакам.

В отличие от дерева-классификатора, дерево-регрессор получается достаточно глубоким, чтобы обеспечить должную точность прогноза. Выбор оптимального набора параметров аналогичен уже рассмотренным алгоритмам – производится поиск по сетке гиперпараметров. Несмотря на некоторую сложность интерпретации именно регрессионного дерева решений, оно, как и дерево-классификатор, показывает хороший результат.

Итоги этапа прогнозирования на основе набора признаков

В целом, все рассмотренные алгоритмы показали хорошие результаты в плане коэффициента детерминации, что говорит о том, что они применимы для поставленной задачи и успешно решают. Однако, метод обычной линейной регрессии внушает меньше доверия, так как несмотря на свою простоту, имеет самую высокую метрику среди рассмотренных алгоритмов, что может говорить о возможном наличии переобучения. Функции *xgb_forecasting()* и *sklearn_forecasting()* реализуют рассмотренные выше алгоритмы и необходимые формальные действия. Первая из них выводит спрогнозированную величину вместе с метрикой алгоритма, а вторая выводит как и массив метрик рассмотренных методов, так и соответствующие предсказания, выполненные каждым из них. Помимо этого *sklearn_forecasting()* выводит и среднее предсказание. Таким образом, пользователю предоставляется максимально широкий набор информации.

Прогнозирование каждого из признаков на основе их предыдущих значений

Для того, чтобы спрогнозировать стоимость доллара в рублях на следующий день, необходимо сначала спрогнозировать каждый из признаков. Будем считать, что все они независимы между собой, то есть предсказание каждого из них будет строиться лишь на основе предыдущих значений соответствующего признака. На данном этапе будут использоваться два похода: статистический и машинное обучение. Второй из них практически не отличается от рассмотренного только что этапа, поэтому изучим сначала первый.

Прогнозирование признаков с помощью ARIMA

Каждый столбец матрицы X (то есть набор значений признака за определенный период) является временным рядом; $x_1, \dots, x_n \in R$, $(x_1, \dots, x_m)$ – столбец указанной матрицы. Ставится задача нахождения функции f_T:

$$y_{T+d} \approx f_T(y_T, \dots, y_1, d) \equiv \hat{y}_{T+d|T}$$

где $d \in 1, \dots, D$ – отсрочка прогноза, D – горизонт прогнозирования. Существуют различные, простые в реализации, модели прогнозирования: скользящее среднее, взвешенное среднее, экспоненциальное сглаживание и др. Мы остановим свой выбор на модели ARIMA – autoregressive integrated moving average, которая реализована в библиотеке statsmodels. Задается эта модель следующим выражением:

$$ARIMA(p,d,q): \Delta^d X_t = c + \sum_{i=1}^{p} a_i \, \Delta^d X_{t-i} + \sum_{j=1}^{q} b_j \, \epsilon_{t-j} + \epsilon_t$$

Разберем последовательно каждый из трех параметров данной модели. Важным понятием в теории временных рядов является понятие стационарности временного ряда – независимость распределения ряда $y_t, \dots, y_{t+s}$ от рассматриваемого промежутка, $\forall s: [t, t+s] \subseteq [1, T]$. Путем дифференцирования, то есть последовательного взятия разностей соседних элементов, ряд может быть приведен к стационарному (при изначальном отсутствии у него данного свойства):

$$y'_t = y_t - y_{t-1}, \forall t = \overline{2, T}$$

В исходном выражении Δ^d – d-кратное дифференцирование. Проверка ряда на стационарность производится с помощью теста Дики-Фуллера: если ряд, путем дифференцирования, приводится к стационарному, то есть присутствует единичный корень ($a = 1$) в уравнении:

$$y_t - a y_{t-1} = \epsilon_t$$

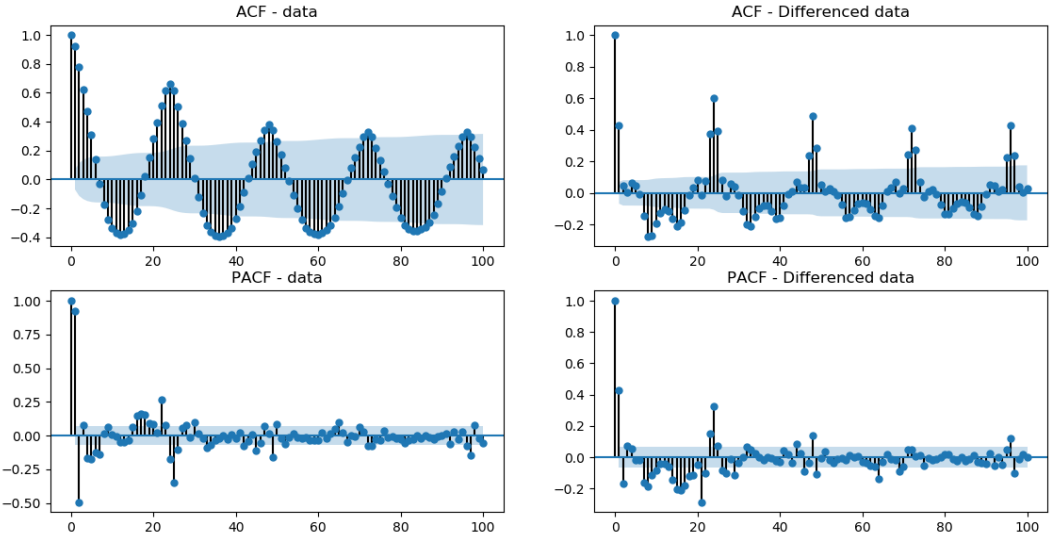
где ϵ_t – ошибка, то исходный ряд считается нестационарным с порядком интегрированности 1 и принимается гипотеза о наличии единичного корня, а если в уравнении выше $|a| < 1$, то данная гипотеза отвергается и ряд считается стационарным. Будем считать, что во всех наших временных рядах имеется и константа, и тренд, поэтому тест можно записать следующим образом:

$$\Delta y_t = b_0 + b_1 t + b y_{t-1} + \epsilon_t$$

где $\Delta y_t = y_t - y_{t-1}$. Для выявления наличия единичного корня считается *t*-статистика:

$$t = \frac{\hat{a} - a}{S_{\hat{a}}}$$

где â – оценка авторегрессионного коэффициента, а $S_{\hat{a}}$ – стандартная ошибка данной оценки, и сравнивается с критическими значениями Дики-Фуллера (Маккинона). Если вычисленная статистика оказывается больше критического значения, то принимается гипотеза о наличии единичного корня, иначе отвергается. Вообще говоря, указанная формула является общей, на самом же деле используется более сложное выражение с использованием винеровских процессов, поэтому не будем углубляться в данный вопрос, так как он уже относится к области эконометрики. Данный алгоритм реализован в функции *adfuller()*, которая находится в библиотеке statsmodels. Порядок интегрированности временного ряда обозначается переменной *d*. Следующие два параметра, которые нам нужно определить – это p и q, коэффициенты частичной автокорреляции и обычной корреляции соответственно. Для каждого из естественно существует формула, но не будем углубляться в теорию временных рядов, а воспользуемся готовым решением из библиотеки statsmodels. Чтобы определить максимально возможные коэффициенты p и q, необходимо посмотреть на графики PACF и ACF и определить число резко выделяющихся точек:



Таким образом, мы получили набор значений p и q (в диапазоне от нуля до соответствующих чисел) и d. Далее, для отбора оптимальной модели, нужно обучить каждую из возможных моделей на ~85% имеющихся данных, проверить их точность на оставшихся ~15% данных, и отобрать ту модель, у которой метрика будет наилучшей. Было принято решение использовать всё тот же коэффициент детерминации. Самым важным этапом, после нахождения параметров, является выбор оптимального периода для обучения, потому что после ряда испытаний было замечено, что выбросы в течение временного интервала или же необъяснимый рост/спад того или иного показателя могут существенно повлиять на то, как модель обучится и “уловит” происходящий процесс. Поэтому, в отличие от третьего этапа, где для нас были важны все имеющиеся данные, для ARIMA набор обучающей сокращен на 100 значений с начала и 230 значений с конца. Данный набор является самым оптимальным из тех, что я пробовал со средней метрикой R^2 около 0.9. Получив прогноз всех признаков на день вперед, можно послать их на вход всем тем моделям их третьего этапа. Все спрогнозированные значения, а также значения некоторых метрик записываются в отдельный файл для удобства пользователя. Выглядит он следующим образом:

```
Forecasted future features: 0.009764867972476707 52.75399232245682 15316.033704414585 83.73205949895639 6630.747312859885 0.0007350840670835078 0.0007360710119932 21 1649.7668714011513

SKLearn models scores and forecasts:
0.9922555237637073 0.902975936817723 0.6806969284637779 0.7400299127126337 0.96141264650594
31.554138312546144 32.37125127783507 32.58578523855397 31.65727537632791 31.701371619047585
SKLearn mean forecast across all models: 31.973964364862137

XGBoost forecasted USD/RUB rate: [31.618397]
```

Несмотря на не идеальные показатели модели ARIMA, прогнозы вышли достаточно точными, со средней ошибкой 30-60 копеек. Самым близким, несмотря на не самую высокую метрику, оказался прогноз с помощью гребневой регрессии. Пользователю предоставляется полная информация по всем прогнозам с метриками, чтобы он мог выбрать максимально подходящий для него (основываясь, быть может, на своем опыте). Рассмотрим теперь прогнозирование признаков средствами, рассмотренными на третьем этапе.

Прогнозирование признаков с помощью SKLearn

Методика, которая используется, практически ничем не отличается от рассмотренной на третьем этапе, за исключением лишь того, что признаками для каждого из признаков будут являться предшествующие значения. Будем считать, что текущее значение признака зависит от 14 предыдущих. Тогда каждая строка матрицы X – очередные

14 значений, а каждый элемент *y* – значение, следующее за теми 14 предыдущими. Такие матрица признаков и столбец ответов составляются для каждого признака и посылаются на вход функции *sklearn_forecasting()*. Полученные результаты очень схожи с теми, что были получены с помощью ARIMA, поэтому данных подход может быть хорошей альтернативой статистическому подходу.

Заключение и выводы

В данной работе была рассмотрена очень актуальная задача – прогнозирование показателей с использованием средств машинного обучения. За каждой из изученных моделей стоит математический аппарат, который также был разобран, чтобы можно было понять принцип обучения. Python и сторонние библиотеки предоставляют широкий функционал для решения как задач регрессии, так и классификации и кластеризации, и всё это удалось испытать в данной работе. Последовательность этапов, изложенная в самом начале, оказалось верной и рабочей, а полученные результаты – прогнозы и методика – оправдали ожидания. Это говорит о том, что написанная программа может быть использована в индустрии. С еще помощью можно делать прогнозы не только в рассмотренной области, но и в других сферах экономики, в производстве и многих других отраслях, где требуется статистический анализ и прогноз для оптимизации того или иного процесса. Данная программа может существовать как и распространяемое ПО, так и веб-сервис, что в наши дни достаточно востребовано. В дальнейшем программа может быть улучшена для повышения точности прогноза, а также дополнена еще одним инструментом машинного обучения – библиотекой *Pycaret*, которую, к сожалению, не довелось попробовать. Для работы пользователю необходимо иметь *.csv* файлы со статистическими данными, занести их в названия в программу, указать временной интервал (изучив график каждого из показателей) и запустить процесс. В ходе работы программы от пользователя лишь потребуется вводить коэффициенты *pacf* и *acf* (после показа соответствующих графиков), остальная работа выполняется автоматически, а записываются в файл отчета.

Источники

- https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D1%8F
- https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B2%D0%B0%D1%80%D0%B8%D0%B0%D1%86%D0%B8%D1%8F
- https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%BC%D0%B5%D0%BD%D1%82%D1%8B_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%BE%D0%B9_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D1%8B
- https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7
- https://m.habr.com/en/post/172043/
- https://habr.com/ru/post/206306/
- https://habr.com/ru/company/ods/blog/327242/
- http://www.machinelearning.ru/wiki/images/d/d0/Psad_ts_arima.pdf
- https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/
- http://www.machinelearning.ru/wiki/index.php?title=%D0%A0%D0%B8%D0%B4%D0%B6-%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F
- http://www.machinelearning.ru/wiki/index.php?title=KNN
- https://habr.com/ru/post/116385/
- https://habr.com/ru/post/314330/