УДК 004

Анализ цены проданных машин на вторичном рынке в Python

Звайгзне Алексей Юрьевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье проводится анализ набора данных по уже проданным автомобилям на вторичном рынке на языке программирования Python, с использованием различных библиотек, для приведения данных необходимых для машинного обучения и построение линейной регрессии, регрессия древа решений, случайного леса, K-ближайших соседей, хребтовая регрессия модели позволяющей определить рыночную цену поддержанного автомобиля сократив расчет стоимости продажи и проверка результатов предсказания с помощью метрик средней квадратичной ошибки, среднеквадратической ошибки и R^2.
Ключевые слова: Python, Scikit-learn, Numpy, Matplotlib, Seaborn, Scipy, Pandas

Analysis of the price of sold machines on the secondary market in Python

Zvaigzne Alexey Yurievich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article analyzes a set of data on cars already sold on the secondary market in the Python programming language, using various libraries, to bring the data necessary for machine learning and the construction of linear regression, regression of the decision tree, random forest, K-nearest neighbors, ridge regression of a model that allows you to determine the market price of a supported car by reducing the cost calculation sales and verification of prediction results using the metrics of mean square error, standard error and R^2.
Keywords: Python, Scikit-learn, Numpy, Matplotlib, Seaborn, Scipy, Pandas

1 Введение

1.1 Актуальность

В настоящее время, с развитием технологий, машинное обучение стало широко использоваться во многих организациях. Эти модели обычно работают с набором предопределенных данных, доступных в виде готовых наборов данных. Зачастую эти наборы данных содержат прошлую/предыдущую информацию о конкретном предмете анализа. Организация этих данных перед их подачей в модель очень важна. Именно

здесь используется анализ данных. Если данные, подаваемые в модель машинного обучения, плохо организованы, модель может выдать ложный ответ или нежелательные выходные данные. Это может привести к серьезным потерям для организации. Следовательно, очень важно использовать надлежащий анализ данных.

1.2 Обзор исследований

Для начала работы в целом необходимо выявить важнейшие параметры авто, зачастую напрямую влияющих на рыночное ценообразование автомобилей [1]. В своей книге К.А. Гадыльшина рассказывает о необходимости применения компьютерной техники для вычисления большого объема данных для уменьшения численной дисперсии [2]. В статье М.Н. Юлдашев выявил критерии оптимизации беспроводных сетей и предложил векторы их развития с точки зрения машинного обучения [3]. В научной статье А. Д. Москвичев анализирует эффективность алгоритмов и возможности их использования в качестве модуля коррелятора в системах класс SIEM [4]. В научной работе А.Д. Варламов предлагал разработку алгоритма оценки близости пар изображений, которые могут применятся в алгоритмах поиска [5]. В сборнике научных трудов Л. Д. Плешков дал первые результаты нового способа автоматической корреляции, указал положительные и отрицательные стороны [6]

1.3 Цель исследования

Данная модель может пригодиться при расчете стоимости машины при последующей продаже её на вторичном рынке. Целью является максимизация прибыли от продажи, но, чтобы при этом цена не была слишком завышенной относительно рынка, для более успешной продажи итоговой машины будущему владельцу автомобиля.

2 Материалы и методы

Для работы с кодом будет использоваться набор данных взятый из репозитория машинного обучения [7]. Для работы с набор данных будет использоваться язык программирования Python, в самом код будут добавлены необходимые для работы с массивами, моделями и графиками библиотеки.

3 Результаты и обсуждения

Данные, которые будут использоваться в этой статье, касаются вторичного рынка продаж автомобилей. В частности, содержит различные информационные данные о подержанных автомобилях, такие как их цена, цвет и т. д. Здесь нужно понимать, что простого сбора данных недостаточно. Исходные данные бесполезны. Здесь анализ данных играет жизненно важную роль в раскрытии необходимости полученной информации и получении нового значения этих необработанных данных.

Для начала необходимо определить, что именно является основной причиной ценообразования автомобилей на вторичном рынке, размер двигателя, цена, цвет, бренд, мощность двигателя или что-то другое?

Исходные данные в данном случае находятся в сыром виде, в форме .data – это обычный текстовой формат, поэтому достаточно поменять расширение пакета на .csv и начать работать.

Для работы понадобятся 6 библиотек:

Pandas – это библиотека с открытым исходным кодом позволяющая выполнять различные действия с данными в Python. Позволяет создавать, обрабатывать и анализировать данные.

NumPy – базовая библиотека позволяющая создавать высокопроизводительные модели машинного обучения схожие с Scikit-learn и Scipy. Используя модели как эффективный многомерный контейнер общих данных.

Matplotlib – это библиотека 2-D графики позволяющая получать в качестве выходных данных графики с отмеченными показателями в различных формах.

Seaborn – более совершенная библиотека визуализации графики в Python, схожа с Matplotlib, но при этом позволяет создавать более привлекательные и понятные графики, отображающие различную статистику.

SciPy – библиотека с открытым исходным кодом, предназначенная для оптимизации, интегрирования, интерполяции, задач на собственные значения, алгебраических уравнений, дифференциальных уравнений, статистики и многих других классов задач.

Scikit-learn– простая и эффективная библиотека для работы с анализом данных, которая может работать с большинством распространенных библиотек по типу NumPy, Scipy и Matplotlib. Работать с различными моделями и предсказывать данные.

Math – библиотека предоставляющая более широкий функционал для работы с числовыми данными.

Все эти библиотеки необходимо предварительно установить в проект непосредственно перед работой с кодом.

После установки библиотек через консоль необходимо их импортировать в самом коде проекта для последующих итераций (рис. 1).

```
#Импортирование необходимых библиотек
from scipy import stats
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, median_absolute_error,
# Для построения моделей линейной регрессии, нахождения
# средней квадратичной ошибки, точности средней квадратичной ошибки
# и общей точности
```

Рисунок 1. Импортирование библиотек

Теперь можно получить данные из файла и проверить правильность массива отобразив первых пять строчек набора (рис. 2).

```
# Использование CSV файла
df = pd.read_csv('data.csv')

# Вывод первых пяти строчек набора данных для проверки
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-syster
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpf
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpf
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpf
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpf
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpf

5 rows × 26 columns

Рисунок 2. Код получения данных из файла и вывод первых пяти строк данных

Данный набор данных из внешнего источника и может содержать пустые значения, поэтому следующим шагом будет проверка набора данных на предмет пропущенных значений если такие есть, путём вывода описательной статистики через команду describe (рис. 3).

```
# Поиск пустых значений или неопределенных значений
df.describe(include="all")
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size
count	205.000000	205	205	205	205	205	205	205	205	205.000000	...	205.000000
unique	NaN	52	22	2	2	3	5	3	2	NaN	...	NaN
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN
freq	NaN	41	32	185	168	114	96	120	202	NaN	...	NaN
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.000000
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	141.000000
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000

11 rows × 26 columns

Рисунок 3. Присваивание заголовков для столбцов

Исходя из выходных данных, видно, что есть поля содержащие неправильные или пустые значения. Некоторые строки содержат знаки вопроса, от них можно сразу избавиться, приведя данные к общему виду (рис. 4).

```
# Избавляемся от знака "?" в данных
df = df.replace('?', np.nan)
df.head()
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-syster
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpf
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpf
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpf
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpf
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpf

5 rows × 26 columns

Рисунок 4. Поиск в наборе данных пустых значений

Так как вопрос, цены конечного автомобиля является ключевым значением, необходимо тщательно его проверить (рис. 5).

```
# Ищем пустые значения в стобце price
df[['price']].head(11)
```

	price
0	13495
1	16500
2	16500
3	13950
4	17450
5	15250
6	17710
7	18920
8	23875
9	NaN
10	16430

Рисунок 5. Вывод столбца «Price»

Видно, что часть данных неопределенна или отсутствует, так как она могла быть договорной и подставляя туда какие-либо значения в итоге можно получить недостоверные данные по итогу, так что лучшим решением в данной ситуации будет просто избавиться от строк с пустыми значениями столбца «Price» (рис. 6).

```
# Удаляем строки с пустыми значениями
df.dropna(subset=["price"], axis=0, inplace=True)
df[["price"]].head(10)
```

	price
0	13495
1	16500
2	16500
3	13950
4	17450
5	15250
6	17710
7	18920
8	23875
10	16430

Рисунок 6. Столбец «Price» без пустых значений

Теперь необходимо в столбцах с целочисленными данными и числами с плавающей точкой, заполнить недостающие значения средними, чтобы сильно не менять общую выборку, пример кода для столбца «Bore» (рис. 7).

```
[7] # Поиск пустых значений в столбце bore
    df[['bore']].loc[53:61]
```

	bore
53	3.03
54	3.08
55	NaN
56	NaN
57	NaN
58	NaN
59	3.39
60	3.39
61	3.39

Рисунок 7. Вывод значений стобца «Bore»

Для того чтобы избавиться в этом столбце от категориальных тип данных (ошибочных), нужно посчитать среднее значение всех ячеек данного столбца и заменить в ошибочных ячейках значения на полученные (рис. 8).

```
[8] # Для параметра Bore нужно вычислить среднее значение
    mean = df[["bore"]].astype(float).mean()
    mean

    bore    3.330711
    dtype: float64

    df[["bore"]] = df[["bore"]].replace(np.nan,mean)
    df[['bore']].loc[53:61]
```

	bore
53	3.03
54	3.08
55	3.330711
56	3.330711
57	3.330711
58	3.330711
59	3.39
60	3.39
61	3.39

Рисунок 8. Замена недостающих значений на среднее по столбцу

Первоначальные данные касательно расхода топлива на милю не совсем удобны для представления и их необходимо преобразовать более понятный «Литры на 100 км» (рис. 9).

```
[ ] df[['city-mpg']]=235/df[['city-mpg']]
    df.rename(columns={'city-mpg' : 'city-L/100km'}, inplace=True)
    df.head(5)
```

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-syster
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpf
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpf
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpf
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpf
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpf

5 rows × 26 columns

Рисунок 9. Преобразование данных и проверка типов данных

Так как тип столбца каркаса базы данных не меняется автоматически, его необходимо изменить вручную (рис. 10).

```
[11] df[['price']].dtypes

     price    object
     dtype: object

[12] # Попытка перевести тип данных цены в число с плавающей точкой
     df[['price']]=df[['price']].astype('float64')
     df[['price']].dtypes

     price    float64
     dtype: object
```

Рисунок 10. Перевод столбца «Price» из категориального в число с плавающей точкой

Теперь можно снова вывести описательную статистику и убедиться, что данные отображаются правильно, для более точной работы с ними (рис. 11)

```
# Повторная проверка данных
df.describe()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-L/100km	
count	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	20
mean	0.840796	98.797015	174.200995	65.889055	53.766667	2555.666667	126.875622	10.164279	9.944145	3
std	1.254802	6.066366	12.322175	2.101471	2.447822	517.296727	41.546834	4.004965	2.534599	
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	4.795918	1
25%	0.000000	94.500000	166.800000	64.100000	52.000000	2169.000000	98.000000	8.600000	7.833333	2
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	9.791667	3
75%	2.000000	102.400000	183.500000	66.600000	55.500000	2926.000000	141.000000	9.400000	12.368421	3
max	3.000000	120.900000	208.100000	72.000000	59.800000	4066.000000	326.000000	23.000000	18.076923	5

Рисунок 11. Повторный вывод описательного анализа данных

Теперь можно проверить столбец «drive-wheels» на наличие неправильных и отсутствующих значений, чтобы не просматривать таблицу целиком можно проверить столбец предмет содержимого, какие категории он содержит с помощью команды value_counts() (рис. 12).

```
# Проверка типа привода автомобилей
drive_whele_count = df['drive-wheels'].value_counts()
drive_whele_count

fwd    118
rwd     75
4wd      8
Name: drive-wheels, dtype: int64
```

Рисунок 12. Вывод количества переменных по значению

С помощью библиотеки seaborn можно вывести боксовый график визуализирующий соотношение цены к типу привода, исходя из этих данных можно определить имеет ли набор данных «Выбросы» (рис. 13).

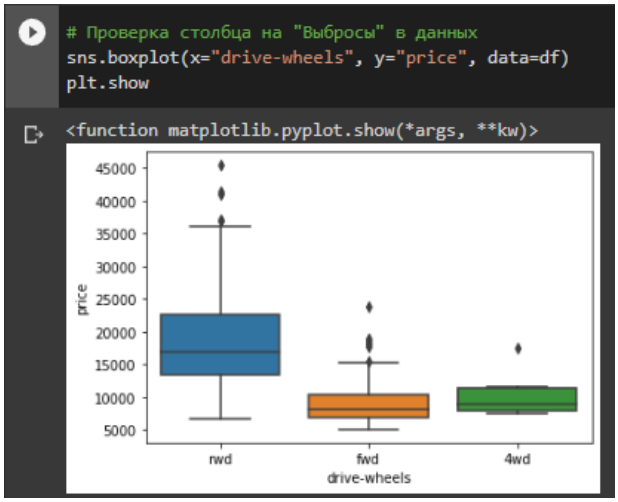

Рисунок 13. Вывод боксового графика

Зачастую цена образовывается исходя из рабочего объема двигателя, так как это является одной из важнейших характеристик автомобиля на рынке, поэтому можно пробовать сразу построить график соотношения рабочего объема двигателя к цене (рис. 14).

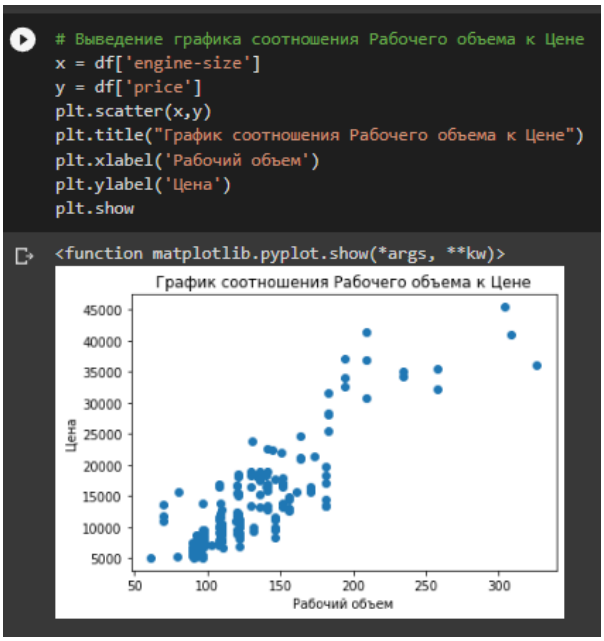

Рисунок 14. Построение графика соотношения рабочего объема двигателя к цене

В большинстве своем машины имеют одинаковую структуру в плане колес, типу кузова и цене, по этим признакам можно группировать эти поля и посмотреть таблицу с ценами по группам (рис. 15).

```
[18] # Выделение уникальных значений в типе привода
     df['drive-wheels'].unique()

     array(['rwd', 'fwd', '4wd'], dtype=object)

[19] # Объединение значений типа привода по виду кузова и цене
     df_group=df[['drive-wheels','body-style', 'price']]

[20] # Выведение средней цены по типу привода
     drive_wheel_avg=df_group.groupby(['drive-wheels'], as_index=False).mean()
     drive_wheel_avg
```

	drive-wheels	price
0	4wd	10241.000000
1	fwd	9244.779661
2	rwd	19757.613333

```
# Выведение средней цены по виду кузова и типу привода
grouped_avg = df_group.groupby(['drive-wheels', 'body-style'], as_index=False).mean()
grouped_avg
```

	drive-wheels	body-style	price
0	4wd	hatchback	7603.000000
1	4wd	sedan	12647.333333
2	4wd	wagon	9095.750000
3	fwd	convertible	11595.000000
4	fwd	hardtop	8249.000000
5	fwd	hatchback	8396.387755
6	fwd	sedan	9811.800000
7	fwd	wagon	9997.333333
8	rwd	convertible	23949.600000
9	rwd	hardtop	24202.714286
10	rwd	hatchback	14337.777778
11	rwd	sedan	21711.833333
12	rwd	wagon	16994.222222

Рисунок 15. Группировка полей и проверка данных

Используя метод поворота, можно отобразить данные согласно соотношению цены, к ранее объединённым типам полей визуализировав данные горизонтально (рис. 16).

```
[22] # Для более простого представления таблицы
     # можно её перевернуть с помощью компанды pivot
     pivot_table = grouped_avg.pivot(index="drive-wheels", columns="body-style")
     pivot_table
```

	price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	NaN	NaN	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222

Рисунок 16. Поворот и вывод таблицы

В полученной таблице, есть неопознаные значения, можно заменить их на нули для более корректного отображения с помощью команды fillna(0) (рис. 17).

```
# Заполнение неопознаных значений 0 и повторный вывод таблицы
pivot_table = pivot_table.fillna(0)
pivot_table
```

	price				
body-style	convertible	hardtop	hatchback	sedan	wagon
drive-wheels					
4wd	0.0	0.000000	7603.000000	12647.333333	9095.750000
fwd	11595.0	8249.000000	8396.387755	9811.800000	9997.333333
rwd	23949.6	24202.714286	14337.777778	21711.833333	16994.222222

Рисунок 17. Замена значений в таблице

Теперь можно вывести тепловую матрицу посмотреть распределние данных более наглядно, чем темнее ячейка, тем ниже значение цены (рис. 18).

```
# Выведение тепловой матрицы
sns.heatmap(pivot_table)
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```

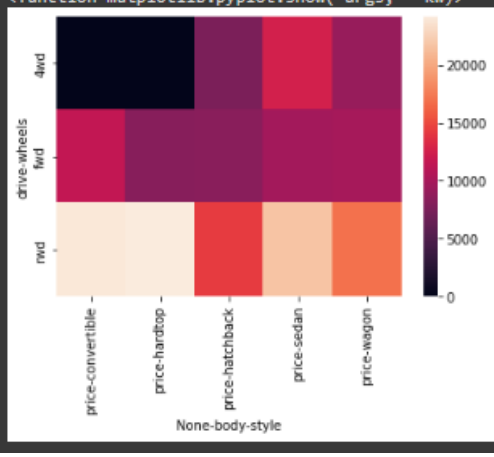

Рисунок 18. Вывод тепловой матрицы

Имея текущие данные, можно провести статистический анализ данных - «The Analysis of Variance (ANOVA) " для этого пригодится библиотека Scipy, чтобы вывести F-тест и P-значение, коррелируя цену и бренд (рис. 19).

```
[29] # Выведение коэфициента и значения корреляции Пирсона для колесной базы
     pearson_coef,p_value=stats.pearsonr(df['wheel-base'],df['price'])
     print("Кэфициент корреляции Пирсона:", pearson_coef, " с P-значениями равными P=", p_value)

     Кэфициент корреляции Пирсона: 0.584641822265508  с P-значениями равными P= 8.076488270733218e-20

[30] # Выведение коэфициента и значения корреляции Пирсона для ширины
     pearson_coef,p_value=stats.pearsonr(df['width'],df['price'])
     print("Кэфициент корреляции Пирсона:", pearson_coef, " с P-значениями равными P=", p_value)

     Кэфициент корреляции Пирсона: 0.7512653440522674  с P-значениями равными P= 9.200335510481516e-38
```

Рисунок 19. Вывод статистики

Так же можно вывести график визуализирующий корреляцию выбрав столбцы Рабочего объема и цены, так как график может быть большим необходимо его ограничить с помощью команды ylim для ограничения графика по оси y (рис. 20).

Рисунок 20. График соотношения Рабочего объема к Цене

Для отображения простой таблицы корреляции, достоточно команды corr() и проверить коэфициенты (рис. 21).

```
# Выведение значений корреляции
df.corr()
```

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-L/100km	highway-mpg	price
symboling	1.000000	-0.535987	-0.365404	-0.242423	-0.550160	-0.233118	-0.110581	-0.182196	0.066171	0.036233	-0.082391
wheel-base	-0.535987	1.000000	0.876024	0.814507	0.590742	0.782097	0.572027	0.250313	0.476153	-0.543304	0.584642
length	-0.365404	0.876024	1.000000	0.857170	0.492063	0.880665	0.685025	0.159733	0.657373	-0.698142	0.690628
width	-0.242423	0.814507	0.857170	1.000000	0.306002	0.866201	0.729436	0.189867	0.673363	-0.680635	0.751265
height	-0.550160	0.590742	0.492063	0.306002	1.000000	0.307581	0.074694	0.259737	0.003811	-0.104812	0.135486
curb-weight	-0.233118	0.782097	0.880665	0.866201	0.307581	1.000000	0.849072	0.156433	0.785353	-0.794889	0.834415
engine-size	-0.110581	0.572027	0.685025	0.729436	0.074694	0.849072	1.000000	0.028889	0.745059	-0.679571	0.872335
compression-ratio	-0.182196	0.250313	0.159733	0.189867	0.259737	0.156433	0.028889	1.000000	-0.299372	0.268465	0.071107
city-L/100km	0.066171	0.476153	0.657373	0.673363	0.003811	0.785353	0.745059	-0.299372	1.000000	-0.930028	0.789898
highway-mpg	0.036233	-0.543304	-0.698142	-0.680635	-0.104812	-0.794889	-0.679571	0.268465	-0.930028	1.000000	-0.704692
price	-0.082391	0.584642	0.690628	0.751265	0.135486	0.834415	0.872335	0.071107	0.789898	-0.704692	1.000000

Рисунок 21. Таблица кореляции данных

Чтобы построить более красивый и понятный график по цветам нужно использовать библиотеку seaborn, выбрать размерность графика и значения отображаемых столбцов, с помощью команды fmt – можно назначить ограничить количество знаков после запятой чтобы избежать переполнения ячеек таблицы (рис. 22).

```
corr=df.corr()
fig, ax = plt.subplots(figsize=(10, 10))
colormap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(corr, cmap=colormap, annot=True, fmt=".2f")
plt.xticks(range(len(corr.columns)), corr.columns)
plt.yticks(range(len(corr.columns)), corr.columns)
plt.show()
```

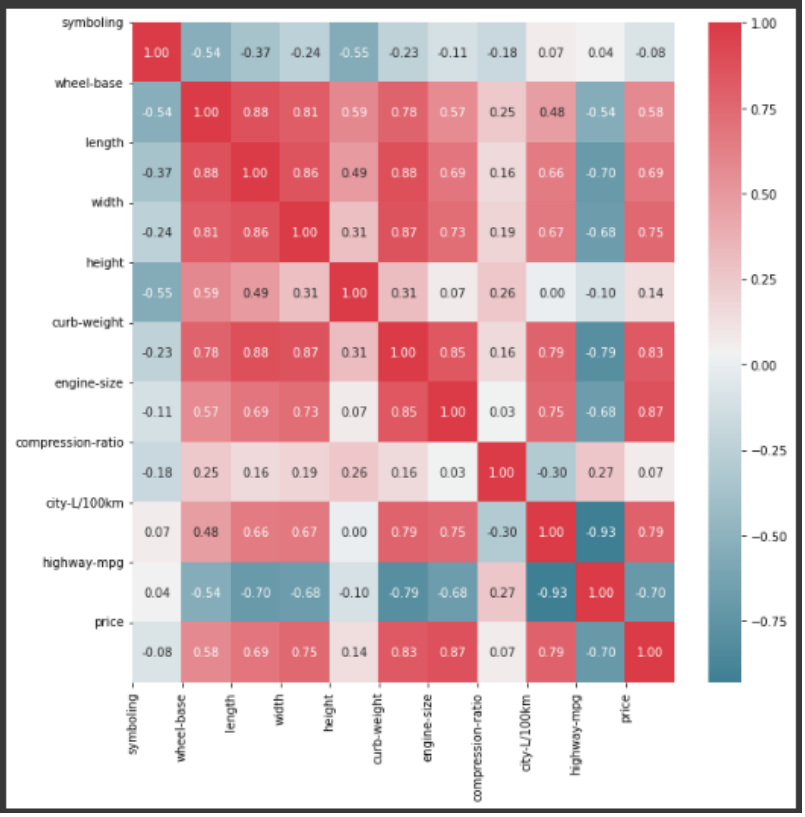

Рисунок 22. График корреляции

Чтобы анализировать набор дальше, необходимо привести данные к определенному виду для создания регрессионной модели и подсчета точности, исключить неиспользуемые данные, исправить ошибочные данные в заполнении, заполнить столбы с числовыми данными средними по столбцу и перевести в число с плавающей точкой, привести из строковых типов к категориальному типу данных.

Для исключения из набора данных неиспользуемые столбцы достаточно команды drop и указать необходимые столбы (рис. 23).

```
# Исключение неиспользуемых данных
df.drop(['make','symboling','normalized-losses'],axis=1,inplace=True)
df.head()
```

	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	length	width	height	...	engine-size
0	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	...	130
1	gas	std	two	convertible	rwd	front	88.6	168.8	64.1	48.8	...	130
2	gas	std	two	hatchback	rwd	front	94.5	171.2	65.5	52.4	...	152
3	gas	std	four	sedan	fwd	front	99.8	176.6	66.2	54.3	...	109
4	gas	std	four	sedan	4wd	front	99.4	176.6	66.4	54.3	...	136

5 rows × 23 columns

Рисунок 23. Исключение неиспользованных данных

Неправдоподобные данные можно заменить с помощью указания точной ячейки, указав подгруппу данных и конкретные значения для замены (рис. 24).

```
df[df['num-of-doors'].isnull()]
df[df['body-style']=='sedan'].value_counts()   #Получение конкретных значений для получения более достоверных данных
df.loc[27, ['num-of-doors']]='four'
df.loc[63, ['num-of-doors']]='four'
```

Рисунок 24. Замена значений в ячейках

Чтобы заменить пустые значения в других столбцах достаточно повторить процедуру аналогичну столбцу «bore» (рис. 25).

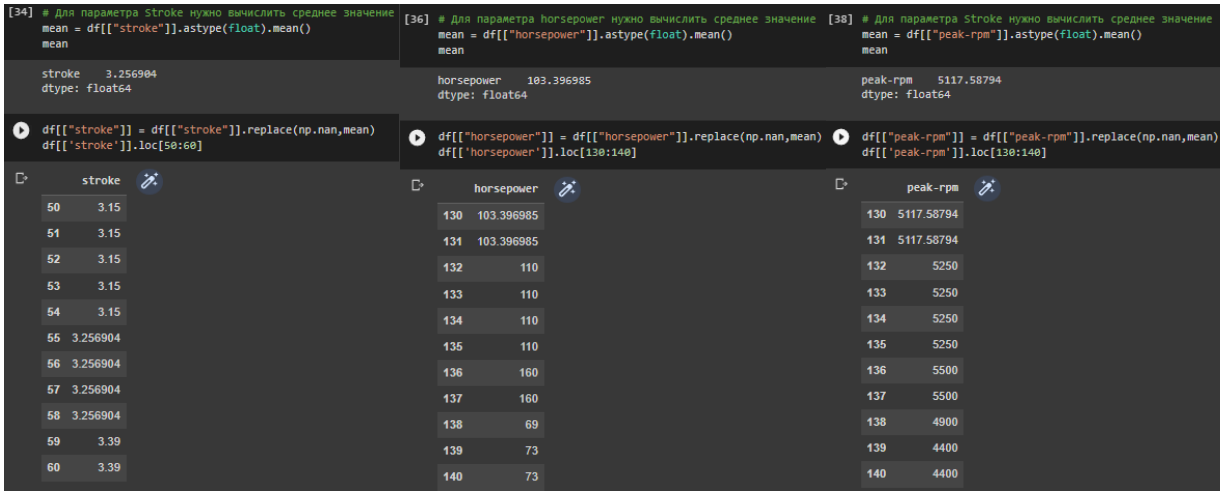

Рисунок 25. Подстановка средних значений заместо пропущенных

Для приведения сразу всех нужных столбцов к категориальному виду можно положить все столбцы в одну переменную и выполнить команду get_dummies из библиотеки pandas, все значения будут приведены к виду 1, 2, 3… и т.д. по категориям (рис. 26).

```
col=['body-style', 'num-of-cylinders','drive-wheels','engine-location','engine-type','fuel-system','num-of-doors','aspiration',
     'fuel-type']
df=pd.get_dummies(df,columns=col,drop_first=True)
```

Рисунок 26. Приведение набора данных к категориальному виду

Теперь можно приступить к созданию модели линейной регрессии, определить тренеровочные и тестовые данные, опытным путем определено, что для получения более точных результов подходит соотношение 20% тестовых данных к 80% тренировочных. Модель основываясь на тренировочных данных сможет предсказать цену автомобиля на тестовых данных (рис. 27).

```
[41] train,test=train_test_split(df,test_size=0.15,random_state=0)

[42] y_train=train.price
     y_test=test.price
     train.drop('price',axis=1,inplace=True)
     test.drop('price',axis=1,inplace=True)

     regressor=LinearRegression()
     model = regressor.fit(train,y_train)
     model.intercept_, model.coef_
     r2 = model.score(train,y_train)
     y_train_pred=model.predict(train)
     y_test_pred=regressor.predict(test)
```

Рисунок 27. Оформление линейной регрессионной модели

Данный анализ набора данных может в будущем упростить процесс определения цены поддержанного автомобиля основывая на текущих рыночных ценах, слабой стороной данной модели является то, что необходимо регулярно мониторить актуальные цены автомобилей на множестве площадок и корректировать её по мере изменения цен и добавления новых видов транспорта.

После выполнения команд можно провести проверку и вывести цену и указать процент погрешности (рис. 28).

```
actual_data=np.array(y_test)
for i in range(len(y_pred)):
    expl=((actual_data[i]-y_pred[i])/actual_data[i])*100.0
    print('Actual Value ${:,.2f},Predicted value ${:,.2f} (%{:,.2f})'.format(actual_data[i],y_pred[i],expl))

Actual Value $6,295.00,Predicted value $5,879.81 (%6.60)
Actual Value $10,698.00,Predicted value $13,492.86 (%-26.13)
Actual Value $13,860.00,Predicted value $14,873.20 (%-7.31)
Actual Value $13,499.00,Predicted value $17,658.23 (%-30.81)
Actual Value $15,750.00,Predicted value $17,983.38 (%-14.18)
Actual Value $8,495.00,Predicted value $9,873.77 (%-16.23)
Actual Value $15,250.00,Predicted value $17,218.57 (%-12.91)
Actual Value $5,348.00,Predicted value $6,118.87 (%-14.41)
Actual Value $21,105.00,Predicted value $19,934.45 (%5.55)
Actual Value $6,938.00,Predicted value $7,681.36 (%-10.71)
Actual Value $11,245.00,Predicted value $9,747.00 (%13.32)
Actual Value $37,028.00,Predicted value $37,338.69 (%-0.84)
Actual Value $7,995.00,Predicted value $11,153.54 (%-39.51)
Actual Value $7,898.00,Predicted value $9,060.30 (%-14.72)
Actual Value $14,869.00,Predicted value $13,755.57 (%7.49)
Actual Value $18,920.00,Predicted value $19,115.41 (%-1.03)
Actual Value $7,129.00,Predicted value $7,183.01 (%-0.76)
Actual Value $15,040.00,Predicted value $24,293.25 (%-61.52)
Actual Value $9,095.00,Predicted value $9,523.93 (%-4.72)
Actual Value $6,189.00,Predicted value $7,091.61 (%-14.58)
Actual Value $9,495.00,Predicted value $11,929.69 (%-25.64)
Actual Value $11,694.00,Predicted value $10,402.29 (%11.05)
Actual Value $35,550.00,Predicted value $32,491.92 (%8.60)
Actual Value $8,058.00,Predicted value $8,804.21 (%-9.26)
Actual Value $10,795.00,Predicted value $12,492.13 (%-15.72)
Actual Value $32,528.00,Predicted value $34,028.00 (%-4.61)
Actual Value $7,975.00,Predicted value $10,284.47 (%-28.96)
Actual Value $11,595.00,Predicted value $13,062.67 (%-12.66)
Actual Value $22,018.00,Predicted value $17,904.24 (%18.68)
Actual Value $32,250.00,Predicted value $32,491.92 (%-0.75)
Actual Value $36,880.00,Predicted value $33,959.39 (%7.92)
Actual Value $15,645.00,Predicted value $14,373.29 (%8.13)
Actual Value $7,898.00,Predicted value $6,338.96 (%19.74)
Actual Value $17,075.00,Predicted value $15,052.03 (%11.85)
Actual Value $7,957.00,Predicted value $10,674.40 (%-34.15)
Actual Value $12,290.00,Predicted value $10,538.85 (%14.25)
Actual Value $12,170.00,Predicted value $14,408.51 (%-18.39)
Actual Value $17,450.00,Predicted value $17,111.80 (%1.94)
Actual Value $8,189.00,Predicted value $10,234.00 (%-24.97)
Actual Value $12,440.00,Predicted value $12,566.99 (%-1.02)
Actual Value $5,118.00,Predicted value $7,095.38 (%-38.64)
```

Рисунок 28. Отображение предсказанных значений и погрешности

Осталось посчитать точность модели используя метрики: средняя квадратичная ошибка, среднеквадратическая ошибка, R^2, для тестовых и тренировочных данных (рис. 29).

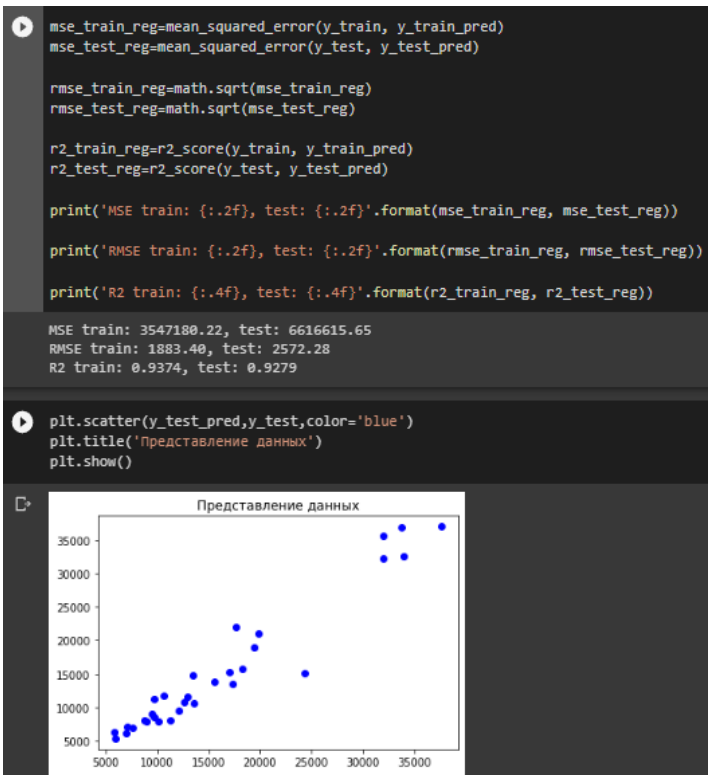

Рисунок 29. Вывод показателей точности и графика предсказанных данных

Теперь для сравнения можно взять другие методы машинного обучения и сравнить полученный результат используя другие методы.

Реализация регрессии древа решений выглядит следующим образом (рис. 30).

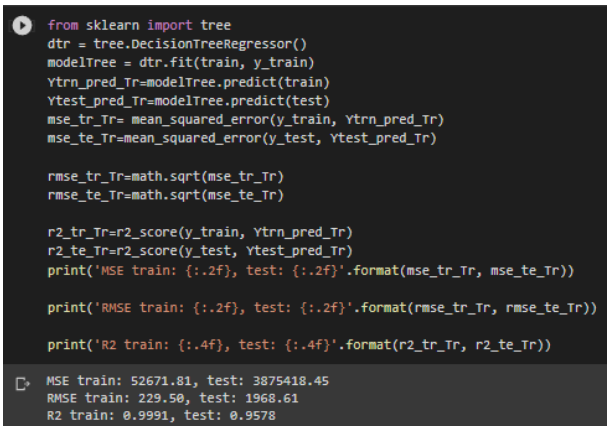

Рисунок 30. Регрессия древа решений с выводом точности

Также у каждого метода можно посмотреть предсказанные данные график отображающий предсказанные данные в соотношении к реальным (рис. 31).

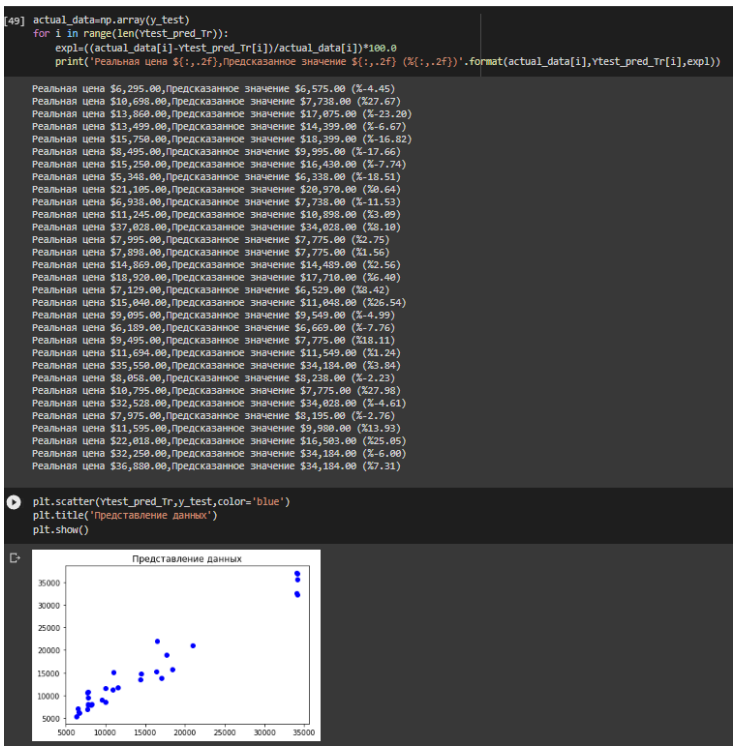

Рисунок 31. Вывод соотношения реальных значений и предсказанных для метода Регрессия древа решений

Данный метод представляет задачу в виде диаграммы, отображающей варианты действий, которые могут быть предприняты в конкретной ситуации и возможные исходы (результаты) каждого из них.

Следующий метод является ансамблем, т.е. суммированием результатов в данном случае, регрессионной модели и метода случайного леса (рис. 32).

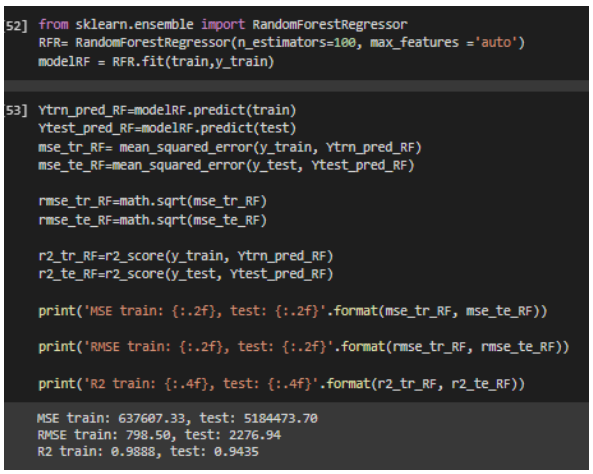

Рисунок 32. Результаты ансамбля Случайного леса и регрессии

Исходя из полученной модели можно также предсказать значения, сравнить их с реальными и вывести график соотношения (рис. 33).

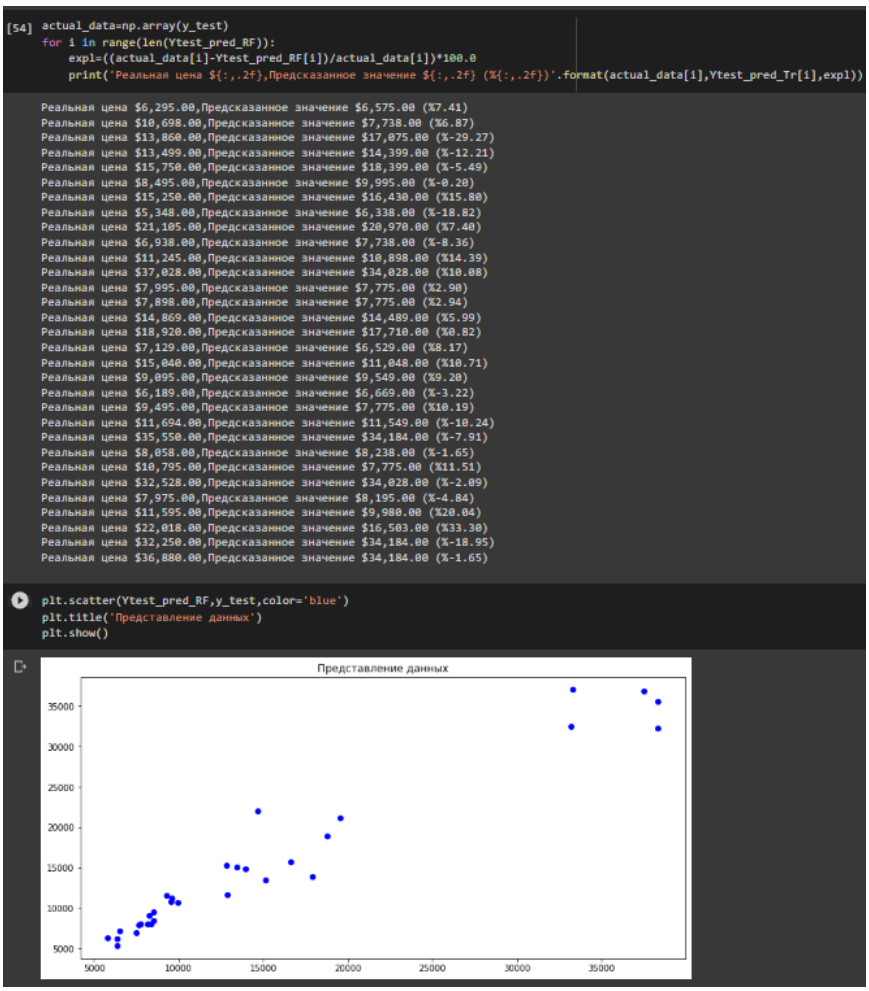

Рисунок 33. Вывод предсказанных данных и графика соотношения предсказанных данных с реальными

С помощью k-ближайших соседей можно выяснить сходство уже известных данных, главным недостатком такого метода является необходимость большого набора тренировочных данных (рис. 34-35).

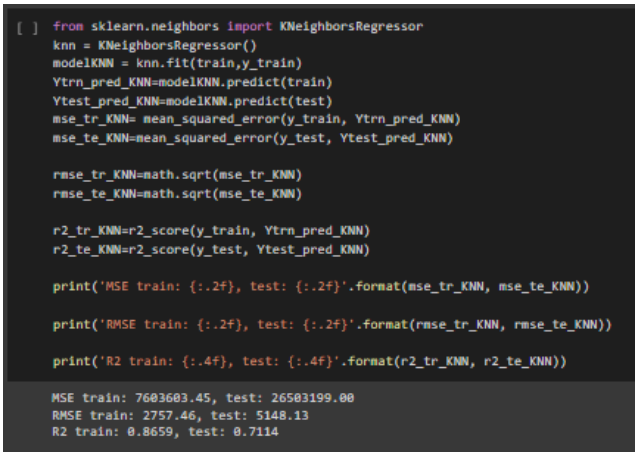

Рисунок 34. Реализация метода k-ближайших соседей

```
actual_data=np.array(y_test)
for i in range(len(Ytest_pred_KNN)):
    expl=((actual_data[i]-Ytest_pred_KNN[i])/actual_data[i])*100.0
    print('Реальная цена ${:,.2f},Предсказанное значение ${:,.2f} (%{:,.2f})'.format(actual_data[i],Ytest_pred_KNN[i],expl))
```

```
Реальная цена $6,295.00,Предсказанное значение $5,899.00 (%6.29)
Реальная цена $10,698.00,Предсказанное значение $10,348.60 (%3.27)
Реальная цена $13,860.00,Предсказанное значение $21,130.60 (%-52.46)
Реальная цена $13,499.00,Предсказанное значение $16,897.20 (%-25.17)
Реальная цена $15,750.00,Предсказанное значение $16,897.20 (%-7.28)
Реальная цена $8,495.00,Предсказанное значение $9,370.20 (%-10.30)
Реальная цена $15,250.00,Предсказанное значение $10,820.40 (%29.05)
Реальная цена $5,348.00,Предсказанное значение $6,556.20 (%-22.59)
Реальная цена $21,105.00,Предсказанное значение $17,794.40 (%15.69)
Реальная цена $6,938.00,Предсказанное значение $7,778.00 (%-12.11)
Реальная цена $11,245.00,Предсказанное значение $9,238.60 (%17.84)
Реальная цена $37,028.00,Предсказанное значение $19,877.60 (%46.32)
Реальная цена $7,995.00,Предсказанное значение $7,658.40 (%4.21)
Реальная цена $7,898.00,Предсказанное значение $7,598.80 (%3.79)
Реальная цена $14,869.00,Предсказанное значение $14,577.00 (%1.96)
Реальная цена $18,920.00,Предсказанное значение $16,993.00 (%10.18)
Реальная цена $7,129.00,Предсказанное значение $7,173.80 (%-0.63)
Реальная цена $15,040.00,Предсказанное значение $11,974.60 (%20.38)
Реальная цена $9,095.00,Предсказанное значение $12,078.00 (%-32.80)
Реальная цена $6,189.00,Предсказанное значение $6,205.60 (%-0.27)
Реальная цена $9,495.00,Предсказанное значение $8,812.80 (%7.18)
Реальная цена $11,694.00,Предсказанное значение $10,165.00 (%13.08)
Реальная цена $35,550.00,Предсказанное значение $38,320.00 (%-7.79)
Реальная цена $8,058.00,Предсказанное значение $8,035.60 (%0.28)
Реальная цена $10,795.00,Предсказанное значение $9,238.60 (%14.42)
Реальная цена $32,528.00,Предсказанное значение $19,194.60 (%40.99)
Реальная цена $7,975.00,Предсказанное значение $8,930.40 (%-11.98)
Реальная цена $11,595.00,Предсказанное значение $9,297.40 (%19.82)
Реальная цена $22,018.00,Предсказанное значение $17,045.00 (%22.59)
Реальная цена $32,250.00,Предсказанное значение $38,320.00 (%-18.82)
Реальная цена $36,880.00,Предсказанное значение $24,914.40 (%32.44)
```

```
plt.scatter(Ytest_pred_KNN,y_test,color='blue')
plt.title('Представление данных')
plt.show()
```

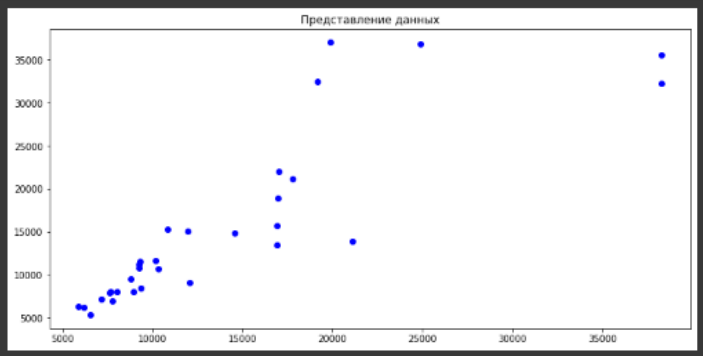

Рисунок 35. Вывод предсказанных данных

С помощью Хребтовой регрессии можно создать модель основывающюся на переменных которые сильно между собой коррелируют, данный метод используют во многих областях, включая химию, экономику, энженерию (рис. 36-37).

```
from sklearn import linear_model
reg = linear_model.Ridge(alpha=.5)
modelBR = reg.fit(train,y_train)
Ytrn_pred_BR=modelBR.predict(train)
Ytest_pred_BR=modelBR.predict(test)
mse_tr_BR= mean_squared_error(y_train, Ytrn_pred_BR)
mse_te_BR=mean_squared_error(y_test, Ytest_pred_BR)

rmse_tr_BR=math.sqrt(mse_tr_BR)
rmse_te_BR=math.sqrt(mse_te_BR)

r2_tr_BR=r2_score(y_train, Ytrn_pred_BR)
r2_te_BR=r2_score(y_test, Ytest_pred_BR)

print('MSE train: {:.2f}, test: {:.2f}'.format(mse_tr_BR, mse_te_BR))

print('RMSE train: {:.2f}, test: {:.2f}'.format(rmse_tr_BR, rmse_te_BR))

print('R2 train: {:.4f}, test: {:.4f}'.format(r2_tr_BR, r2_te_BR))
```

```
MSE train: 3885821.65, test: 5432650.80
RMSE train: 1971.25, test: 2330.80
R2 train: 0.9315, test: 0.9408
```

Рисунок 36. Реализация хребтовой регрессии

```
actual_data=np.array(y_test)
for i in range(len(Ytest_pred_BR)):
    exp1=((actual_data[i]-Ytest_pred_BR[i])/actual_data[i])*100.0
    print('Реальная цена ${:,.2f},Предсказанное значение ${:,.2f} (%{:,.2f})'.format(actual_data[i],Ytest_pred_BR[i],exp1))
```

```
Реальная цена $6,295.00,Предсказанное значение $5,768.25 (%8.37)
Реальная цена $10,698.00,Предсказанное значение $13,007.50 (%-21.59)
Реальная цена $13,860.00,Предсказанное значение $15,276.67 (%-10.22)
Реальная цена $13,499.00,Предсказанное значение $18,560.56 (%-37.50)
Реальная цена $15,750.00,Предсказанное значение $18,620.42 (%-18.22)
Реальная цена $8,495.00,Предсказанное значение $9,999.44 (%-17.71)
Реальная цена $15,250.00,Предсказанное значение $17,396.92 (%-14.08)
Реальная цена $5,348.00,Предсказанное значение $5,815.82 (%-8.75)
Реальная цена $21,105.00,Предсказанное значение $19,955.97 (%5.44)
Реальная цена $6,938.00,Предсказанное значение $7,390.94 (%-6.53)
Реальная цена $11,245.00,Предсказанное значение $9,876.20 (%12.17)
Реальная цена $37,028.00,Предсказанное значение $33,418.69 (%9.75)
Реальная цена $7,995.00,Предсказанное значение $11,323.21 (%-41.63)
Реальная цена $7,898.00,Предсказанное значение $8,597.09 (%-8.85)
Реальная цена $14,869.00,Предсказанное значение $13,718.52 (%7.74)
Реальная цена $18,920.00,Предсказанное значение $19,759.28 (%-4.44)
Реальная цена $7,129.00,Предсказанное значение $6,767.79 (%5.07)
Реальная цена $15,040.00,Предсказанное значение $21,018.27 (%-39.75)
Реальная цена $9,095.00,Предсказанное значение $9,234.32 (%-1.53)
Реальная цена $6,189.00,Предсказанное значение $6,895.07 (%-11.41)
Реальная цена $9,495.00,Предсказанное значение $11,751.91 (%-23.77)
Реальная цена $11,694.00,Предсказанное значение $10,058.69 (%13.98)
Реальная цена $35,550.00,Предсказанное значение $32,989.10 (%7.20)
Реальная цена $8,058.00,Предсказанное значение $8,595.51 (%-6.67)
Реальная цена $10,795.00,Предсказанное значение $12,529.05 (%-16.06)
Реальная цена $32,528.00,Предсказанное значение $31,436.09 (%3.36)
Реальная цена $7,975.00,Предсказанное значение $10,266.65 (%-28.74)
Реальная цена $11,595.00,Предсказанное значение $12,331.36 (%-6.35)
Реальная цена $22,018.00,Предсказанное значение $17,916.63 (%18.63)
Реальная цена $32,250.00,Предсказанное значение $32,989.10 (%-2.29)
Реальная цена $36,880.00,Предсказанное значение $33,494.71 (%9.18)
```

```
[61] plt.scatter(Ytest_pred_BR,y_test,color='blue')
     plt.title('Представление данных')
     plt.show()
```

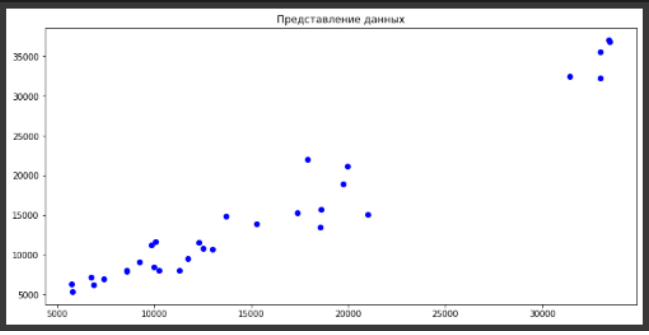

Рисунок 37. Вывод результатов работы модели

Выводы

По итогу получено пять моделей, линейной регрессии, регрессия древа решений, случайного леса, k-ближайших соседей и хребтовая регрессия, а также посчитаны результаты их точности с помощью трех видов метрик средней квадратичной ошибки, среднеквадратической ошибки и R^2.

Для сравнительного анализа точности предсказанных можно вывести таблицу отображающую среднеквадратическую ошибку и R^2 для тренировочных данных и тестовых, а также сравнить полученные результаты (рис. 38).

```
mes = pd.DataFrame(np.array([['RMSE Train',mse_train_reg,rmse_tr_Tr,rmse_tr_RF,rmse_tr_KNN,rmse_tr_BR],
                             ['RMSE Test', mse_test_reg,rmse_te_Tr,rmse_te_RF,rmse_te_KNN,rmse_te_BR],
                             ['R2 Train', r2_train_reg,r2_tr_Tr,r2_tr_RF,r2_tr_KNN,r2_tr_BR],
                             ['R2 Test', r2_test_reg,r2_te_Tr,r2_te_RF,r2_te_KNN,r2_te_BR]]),
                    columns=['Метрики','Линейная Регрессия','Древо решений','Случайный лес','KNN','Хребет'])
mes=mes.set_index('Метрики')
mes
```

Метрики	Линейная Регрессия	Древо решений	Случайный лес	KNN	Хребет
RMSE Train	3547180.2201446984	229.50338969686905	798.5031786609181	2757.4632275079975	1971.2487551621273
RMSE Test	6616615.650039611	1968.6082524496599	2276.9439379059972	5148.125775714949	2330.804753634176
R2 Train	0.9374303966020037	0.9990709087783026	0.9887530841242126	0.8658781277515751	0.9314570152283457
R2 Test	0.927945493950469	0.9577969498251672	0.9435414249403874	0.7113819188134437	0.9408387927293606

Рисунок 38. Вывод таблицы с результатами точности

Наименее эффективным на данном наборе данных оказался метод k-ближайших соседей, так как данных в наборе оказалось недостаточно для обучения модели и на тестовых данных это наглядно видно 88% и 71% соответственно. Самым же эффективным оказался метод регрессионного древа решений, на тренировочной модели он получил показатель в 99%, а на тестовых 95%. В целом все модели показали высокие показатели точности.

Библиографический список

1. Статья на тему «Характеристика автомобиля» URL: https://educam.ru/harakteristiki-avtomobilya/ (дата обращения 15.01.2023).
2. Гадыльшина К. А. Применение методов машинного обучения для подавления численной дисперсии в задаче численного моделирования полных волновых сейсмических полей //Интерэкспо Гео-Сибирь. 2021. Т. 2. №. 2. С. 17-25.
3. Юлдашев М. Н. Классификация состояний беспроводной сенсорной сети с использованием методов машинного обучения //Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС). 2016. №. 2. С. 248-251.
4. Москвичев А. Д., Долгачев М. В. Алгоритмы корреляции событий информационной безопасности //Автоматизация процессов управления. 2020. №. 3. С. 50-59.
5. Варламов А. Д., Шарапов Р. В. Поиск визуально подобных изображений на основе машинного обучения //Труды. 2012. С. 113-120.
6. Плешков Л. Д. Автоматическая парная корреляция скважин на основе анализа с применением Марковской модели //Теория и практика разведочной и промысловой геофизики. 2019. С. 225-230.
7. Архив наборов данных для машинного обучения URL: https://archive.ics.uci.edu/ml/datasets.php (Дата обращения 15.01.2023)