

Практическая №17

Multiple Linear Regression

Цель работы: использовать scikit-learn для множественной линейной регрессии, создание, оценивание и использование модели для прогнозирования неизвестного значения.

Multiple Linear Regression (множественная линейная регрессия) — это статистический метод, который используется для моделирования зависимости между одной зависимой переменной (обычно обозначается как y) и несколькими независимыми переменными (обычно обозначаются как x_1, x_2, \dots, x_n). Цель этого метода — найти линейное уравнение, которое наилучшим образом описывает эту зависимость.

Загружаем файл, содержащий показатели расхода топлива для конкретной модели и предполагаемые выбросы углекислого газа для новых малотоннажных автомобилей для розничной продажи в Канаде. Построим график зависимости значений выбросов от объема двигателя:

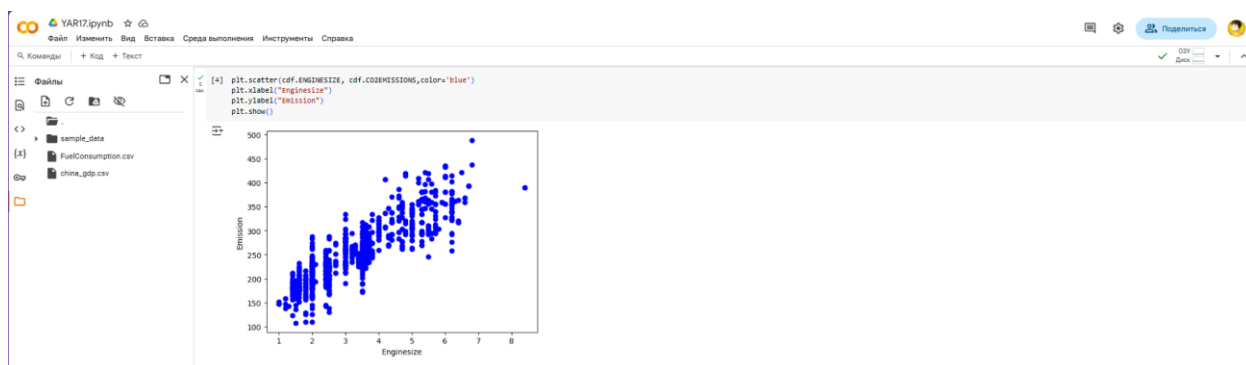


Рисунок 1 - Модель и коэффициенты



Рисунок 1.1 – Данные точности

Далее практическое задание. Нам необходимо изменить один из параметров, на которых строится модель и после оценивается. А конкретно

посмотреть, какие будут значения при «FUELCONSUMPTION_CITY» и «FUELCONSUMPTION_HWY»:

```
[7]: from sklearn import linear_model
regr = linear_model.LinearRegression()
x = np.asarray(train[['ENGINE_SIZE', 'CYLINDERS', 'FUELCONSUMPTION_CITY']])
y = np.asarray(train[['CO2EMISSIONS']])
regr.fit(x, y)
#The coefficients
print('Coefficients: ', regr.coef_)

Coefficients: [[11.99511212  7.24284213  9.33889092]]

[8]: y_hat = regr.predict(test[['ENGINE_SIZE', 'CYLINDERS', 'FUELCONSUMPTION_CITY']])
x = np.asarray(test[['ENGINE_SIZE', 'CYLINDERS', 'FUELCONSUMPTION_CITY']])
y = np.asarray(test[['CO2EMISSIONS']])
print("Residualsumofsquares: %.2f" % np.mean((y_hat - y) ** 2))
#Explainedvariancescore: is perfect prediction
print("Variancescore: %.2f" % regr.score(x, y))

Residualsumofsquares: 1516.68
Variancescore: 0.86

Practice

[9]: # @title Practice
from sklearn import linear_model
regr = linear_model.LinearRegression()
x = np.asarray(train[['ENGINE_SIZE', 'CYLINDERS', 'FUELCONSUMPTION_CITY']])
y = np.asarray(train[['CO2EMISSIONS']])
regr.fit(x, y)
#The coefficients
print('Coefficients: ', regr.coef_)

Coefficients: [[12.42336186  5.97785933  8.22871366]]

Practice

[10]: # @title Practice
y_hat = regr.predict(test[['ENGINE_SIZE', 'CYLINDERS', 'FUELCONSUMPTION_HWY']])
x = np.asarray(test[['ENGINE_SIZE', 'CYLINDERS', 'FUELCONSUMPTION_HWY']])
y = np.asarray(test[['CO2EMISSIONS']])
print("Residualsumofsquares: %.2f" % np.mean((y_hat - y) ** 2))
#Explainedvariancescore: is perfect prediction
print("Variancescore: %.2f" % regr.score(x, y))

Residualsumofsquares: 1713.95
Variancescore: 0.54
```

Рисунок 1.2 - Коэффициенты и новые точности

Исходя из данных с рисунков мы видим, что точность при «городской» переменной немного лучше, в то время как при «шоссе» она заметно хуже.

Вывод: в результате практической работы мы реализовали scikit-learn для множественной линейной регрессией. Создали и оценили собственную модель для прогнозирования неизвестного значения. И в процессе оценки поняли, что лучше проверять все переменные, чтобы найти самую точную.