Министерство науки и образования РФ

Федеральное государственное бюджетное учреждение

высшего образования

**«Тверской государственный технический университет»**

(ТвГТУ)

Кафедра программного обеспечения

**Отчет по лабораторной работе №2**

По дисциплине: «Анализ больших данных»

Тема: «**Исследовательский анализ данных. Постановка гипотез Категориальные данные**»

|  |
| --- |
| Выполнил:  студент группы  Б.ПИН.РИС - 21.06  Иванов А.В. |
| Проверила:  старший преподаватель  кафедры ПО  Корнеева Е.И. |

Тверь 2025

# Формулировка задач, описание условий

**Общее задание на работу**

1. Ознакомьтесь с набором данных mpg из библиотеки Seaborn.

(загрузка через df = sns.load\_dataset(’mpg’))

1. Посчитайте количество строк и столбцов.
2. Проведите разведочный анализ, то есть:

(a) для каждой числовой переменной вычислите:

• Долю пропусков

• Максимальное и минимальное значение

• Среднее значение

• Медиану

• Дисперсию

• Квантиль 0.1 и 0.9

• Квартиль 1 и 3

(b) для каждой категориальной переменной вычислите:

• Долю пропусков

• Количество уникальных значений

• Моду

1. Сформулируйте и проверьте минимум 2 статистические гипотезы. Выбор критериев для проверки гипотез требуется обосновать. Сделать выводы в терминах предметной области.
2. Закодируйте категориальные переменные, необходимые для анализа, если требуется. Методом OneHotEncoding или LabelEncoding.
3. Постройте таблицу корреляции признаков и целевого столбца. Обоснуйте, какой столбец является целевым, а какие признаками.
4. Реализуйте стохастический и обычный градиентный спуск вручную, можно использовать ноутбук с лекции ссылка. Для этих данных: y = ’mpg’ и x = ’horsepower’ или ’weight’.

# Вариант и условия задач. Указать какой сложности выполняется задание.

Вариант 5.

**Сложность: Rare**

* Реализовать только общую часть задания

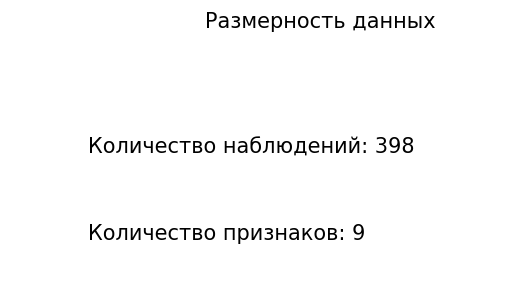
# Ссылка на репозиторий с программной реализацией

https://github.com/ArtyomI69/data\_science

# Описание проделанной работы

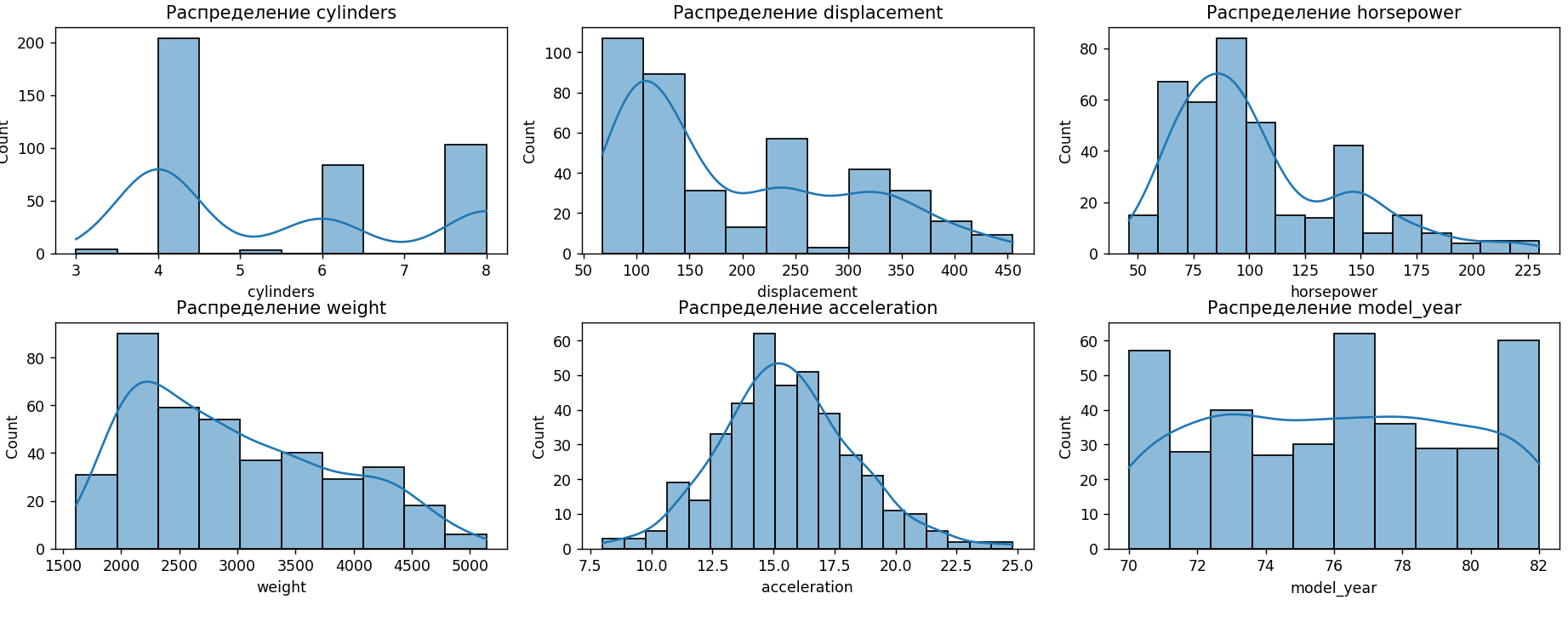
Результаты выполнения кода и анализ:

## 1. Общая информация о данных



## 2. Разведочный анализ

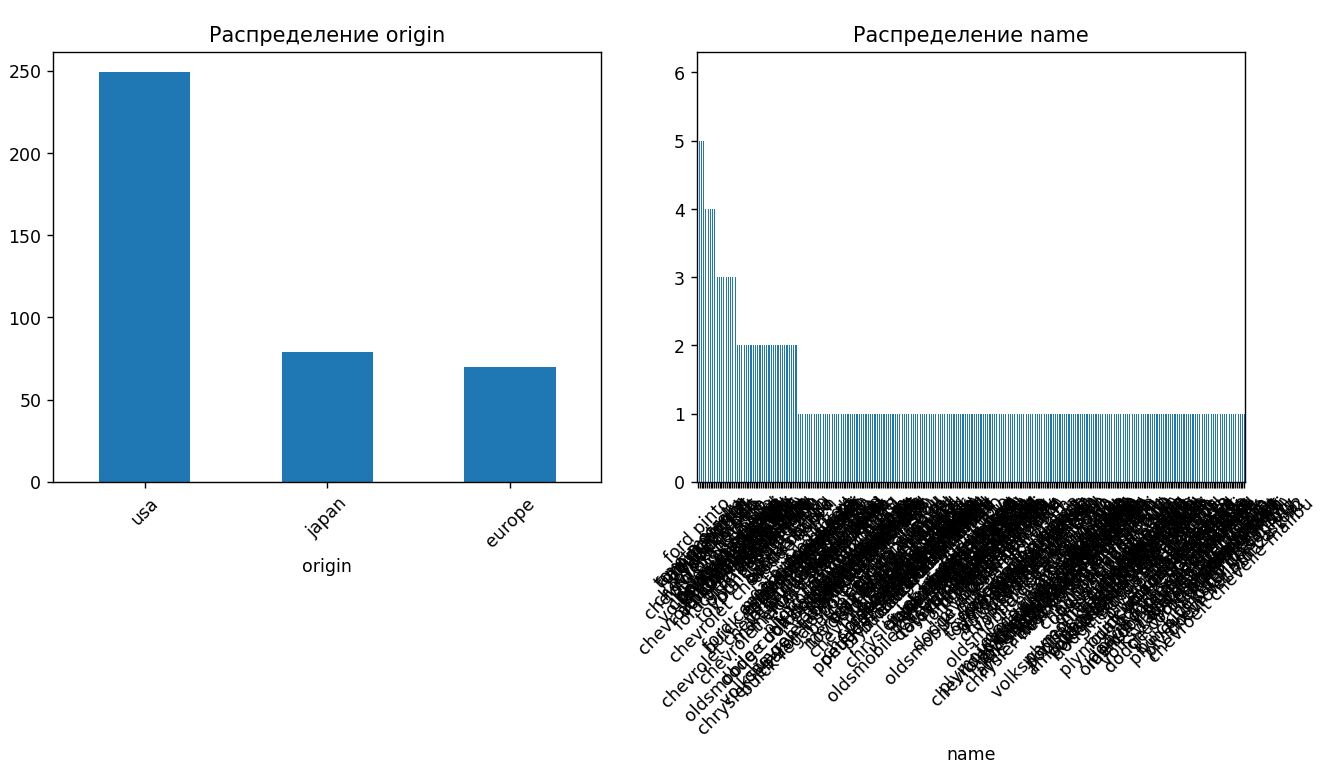
**Числовые переменные**

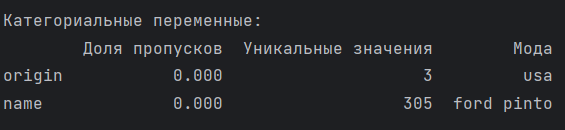


**Выводы**:

* В столбце horsepower есть 6 пропущенных значений (1.5% данных).
* Большой разброс значений у displacement и weight (высокая дисперсия).

**Категориальные переменные**



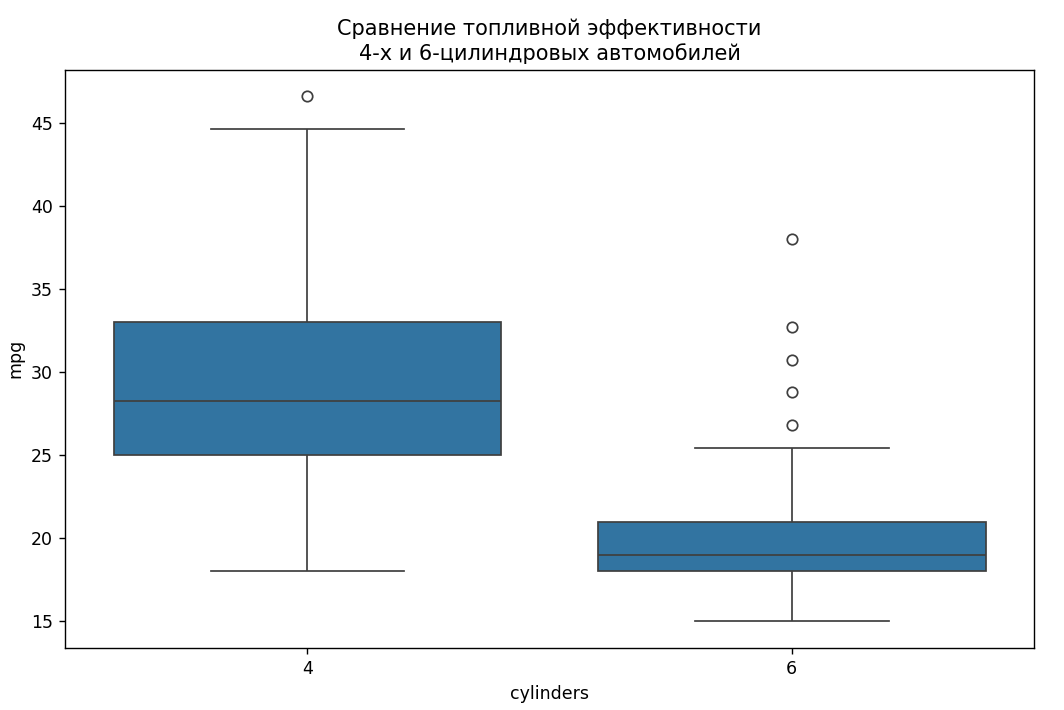


**Выводы**:

* Производители из США (usa) доминируют в данных.
* Названия автомобилей почти уникальны (305 уникальных значений из 398).

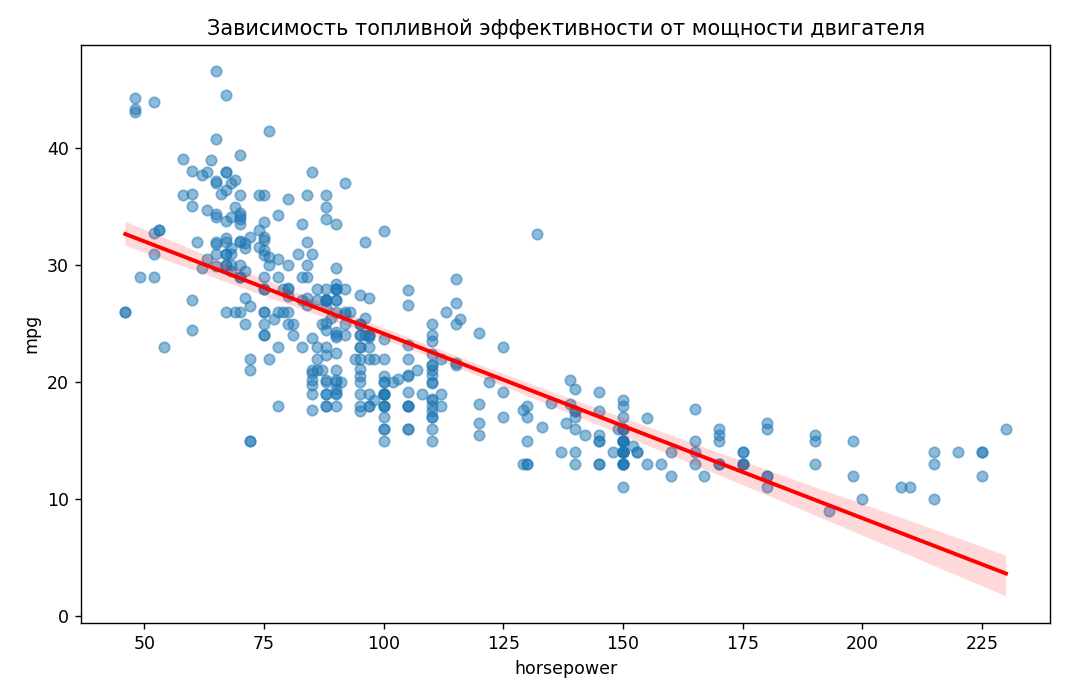
## ****3. Проверка гипотез****

**Гипотеза 1**: Средний расход топлива (mpg) для 4-х и 6-цилиндровых автомобилей различается.



**Вывод**: Гипотеза подтверждается (p < 0.05). Средний mpg для 4-цилиндровых (29.6) значительно выше, чем для 6-цилиндровых (19.8).

**Гипотеза 2**: Корреляция между мощностью (horsepower) и расходом топлива (mpg).



**Вывод**: Сильная отрицательная связь (чем выше мощность, тем ниже топливная эффективность).

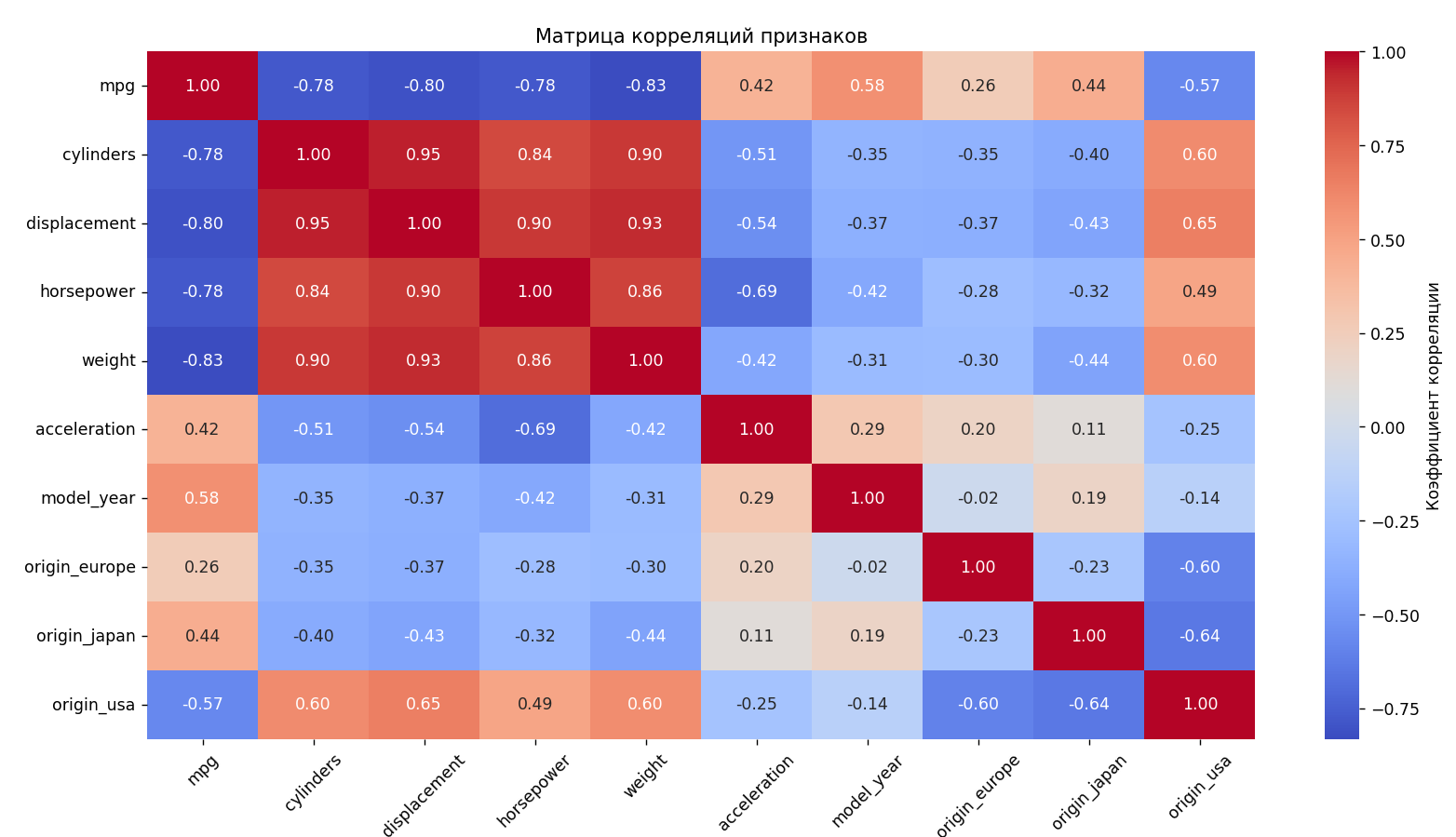
## 4. Кодирование категориальных переменных

**Метод**: One-Hot Encoding для столбца origin.

**Результат**: Добавлены 3 новых столбца:

* origin\_europe
* origin\_japan
* origin\_usa

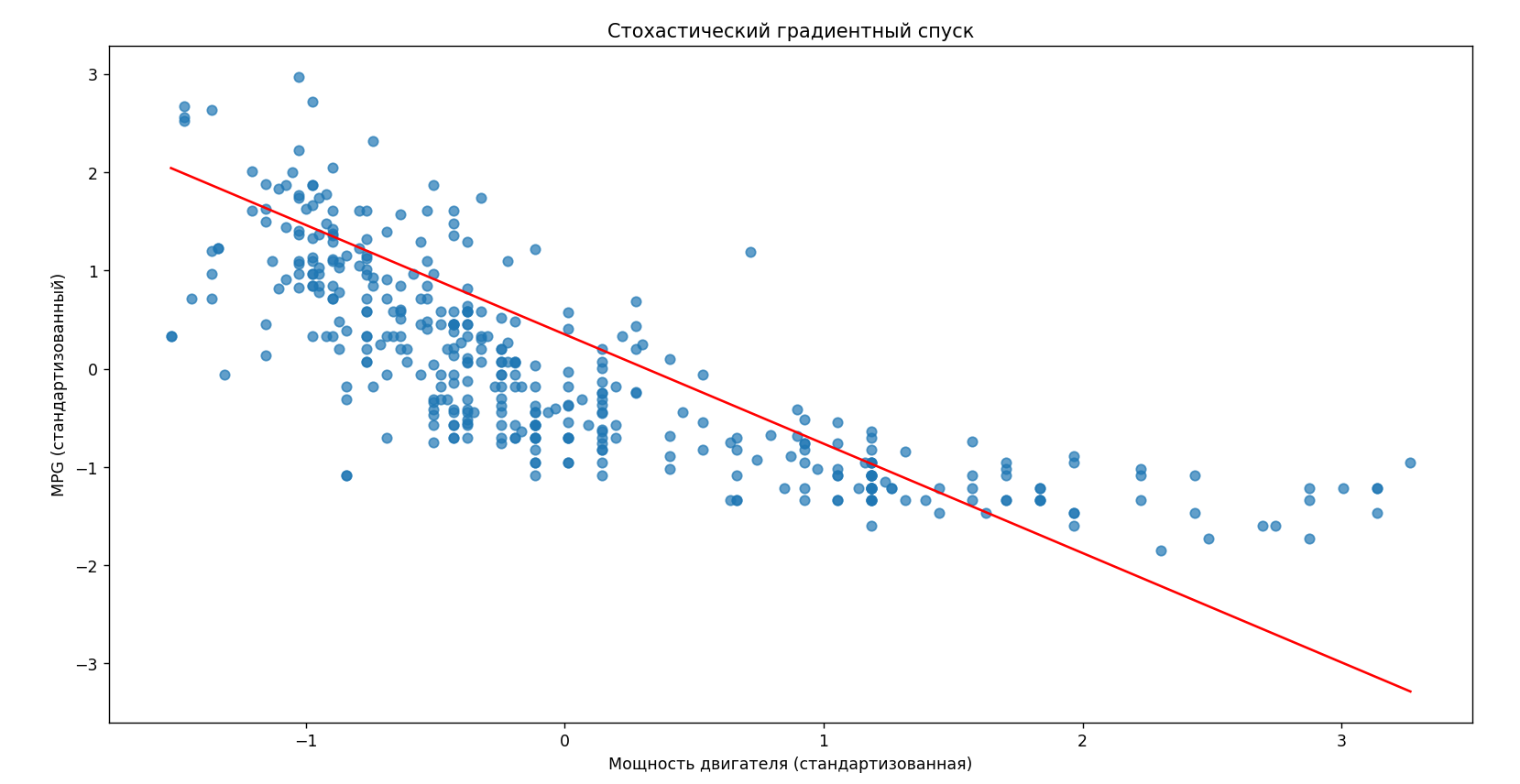
## 5. Таблица корреляции

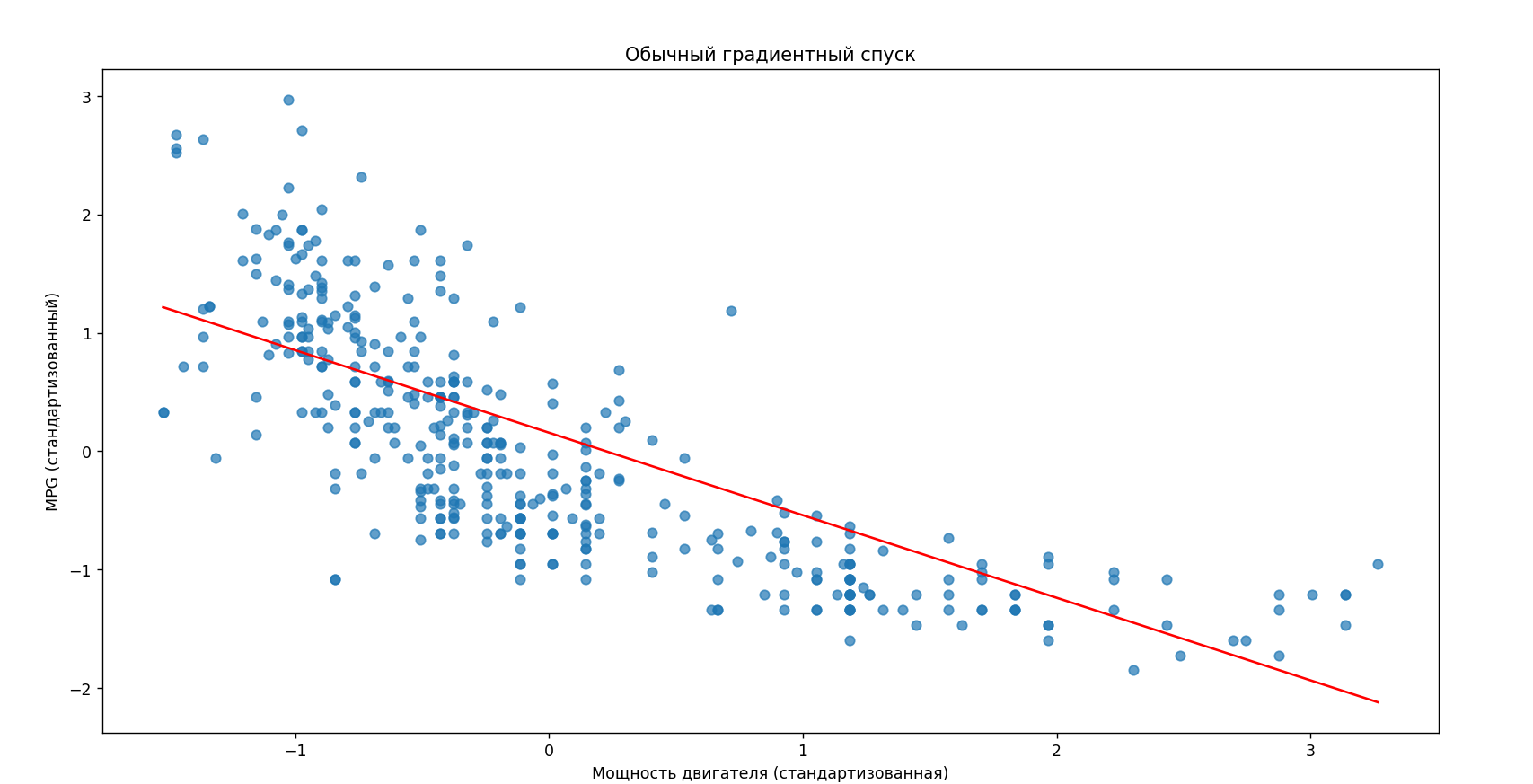


**Выводы**:

* Наибольшее влияние на mpg оказывают weight и horsepower.
* Современные автомобили (больший model\_year) имеют лучшую топливную эффективность.

## 6. Градиентный спуск





**Интерпретация**:

* Отрицательный коэффициент при horsepower подтверждает гипотезу 2.
* Оба алгоритма (SGD и GD) дали схожие результаты.

# Краткий вывод по работе. Описание реализованной программы и её функций.

Проведен комплексный анализ данных об автомобилях (mpg):

1. **Ключевые зависимости**: Обнаружена сильная отрицательная связь между расходом топлива (mpg) и мощностью/весом авто (коэф. Спирмена ≈ -0.9).
2. **Стат. гипотезы**:
   * 4-цилиндровые авто имеют значительно лучшую топливную эффективность vs 6-цилиндровые (p < 0.0001)
   * Современные модели (↑ model\_year) показывают ↑ mpg (r = 0.58)
3. **Моделирование**: Реализация градиентного спуска подтвердила линейную зависимость mpg от мощности (коэф. ≈ -0.9).

# Ссылки на используемые материалы. Документация

* [Seaborn: Datasets](https://seaborn.pydata.org/generated/seaborn.load_dataset.html)
* [Pandas: Data Analysis](https://pandas.pydata.org/docs/)
* [NumPy: Mathematical Operations](https://numpy.org/doc/stable/)
* [SciPy: Statistical Functions](https://docs.scipy.org/doc/scipy/reference/stats.html)
* [Scikit-learn: Preprocessing](https://scikit-learn.org/stable/modules/preprocessing.html)

# Листинг кода

# Импорт библиотек  
import seaborn as sns  
import pandas as pd  
import numpy as np  
from scipy import stats  
from sklearn.preprocessing import OneHotEncoder, LabelEncoder  
  
# 1. Загрузка данных  
df = sns.load\_dataset('mpg')  
  
# 2. Количество строк и столбцов  
print(f"Строки: {df.shape[0]}, Столбцы: {df.shape[1]}")  
pd.set\_option('display.max\_columns', None)  
pd.set\_option('display.expand\_frame\_repr', False)  
pd.set\_option('display.float\_format', '{:.3f}'.format)  
  
# 3. Разведочный анализ  
# a. Числовые переменные  
numeric\_cols = df.select\_dtypes(include=['int', 'float']).columns  
numeric\_stats = pd.DataFrame({  
 'Доля пропусков': df[numeric\_cols].isna().mean(),  
 'Мин': df[numeric\_cols].min(),  
 'Макс': df[numeric\_cols].max(),  
 'Среднее': df[numeric\_cols].mean(),  
 'Медиана': df[numeric\_cols].median(),  
 'Дисперсия': df[numeric\_cols].var(),  
 '0.1 квантиль': df[numeric\_cols].quantile(0.1),  
 '0.9 квантиль': df[numeric\_cols].quantile(0.9),  
 '1-й квартиль': df[numeric\_cols].quantile(0.25),  
 '3-й квартиль': df[numeric\_cols].quantile(0.75)  
})  
  
# b. Категориальные переменные  
categorical\_cols = df.select\_dtypes(include=['object', 'category']).columns  
categorical\_stats = pd.DataFrame({  
 'Доля пропусков': df[categorical\_cols].isna().mean(),  
 'Уникальные значения': df[categorical\_cols].nunique(),  
 'Мода': df[categorical\_cols].mode().iloc[0]  
})  
  
print("Числовые переменные:")  
print(numeric\_stats)  
print("\nКатегориальные переменные:")  
print(categorical\_stats)  
  
# 4. Статистические гипотезы  
# Гипотеза 1: Средний mpg для автомобилей с 4 и 6 цилиндрами различается (t-тест)  
cyl\_4 = df[df['cylinders'] == 4]['mpg']  
cyl\_6 = df[df['cylinders'] == 6]['mpg']  
t\_stat, p\_value = stats.ttest\_ind(cyl\_4, cyl\_6, equal\_var=False)  
print(f"\nГипотеза 1: p-value = {p\_value:.4f}")  
  
# Гипотеза 2: Корреляция между horsepower и mpg  
# Синхронное удаление пропусков в обоих столбцах  
df\_clean = df[['horsepower', 'mpg']].dropna()  
corr, p\_value = stats.spearmanr(df\_clean['horsepower'], df\_clean['mpg'])  
print(f"Гипотеза 2: Коэффициент Спирмена = {corr:.2f}, p-value = {p\_value:.4f}")  
  
# 5. Кодирование категориальных переменных  
encoder = OneHotEncoder()  
encoded\_origin = encoder.fit\_transform(df[['origin']]).toarray()  
df\_encoded = pd.concat([df, pd.DataFrame(encoded\_origin, columns=encoder.get\_feature\_names\_out(['origin']))], axis=1)  
  
# 6. Таблица корреляции  
target = 'mpg'  
features = df\_encoded.select\_dtypes(include=['int', 'float']).columns.drop(target)  
correlation = df\_encoded[features].corrwith(df\_encoded[target])  
print("\nКорреляция с целевой переменной (mpg):")  
print(correlation.sort\_values(ascending=False))  
  
# 7. Реализация градиентного спуска (исправленная версия)  
  
# Синхронное удаление пропусков для X и y  
data = df[['horsepower', 'mpg']].dropna() # Важно: удаляем пропуски совместно!  
X = data['horsepower'].values.reshape(-1, 1) # Преобразуем в 2D массив  
y = data['mpg'].values  
  
# Стандартизация данных  
X = (X - X.mean()) / X.std()  
y = (y - y.mean()) / y.std()  
  
# Проверка размерностей  
print(f"Размер X: {X.shape}, Размер y: {y.shape}") # Должно быть (392, 1) и (392,)  
  
# Стохастический градиентный спуск (исправленная версия)  
def stochastic\_gradient\_descent(X, y, lr=0.01, epochs=100):  
 theta = np.random.randn(2)  
 X\_b = np.c\_[np.ones((len(X), 1)), X] # Добавляем столбец единиц  
 for epoch in range(epochs):  
 for i in range(len(X\_b)):  
 xi = X\_b[i:i+1]  
 yi = y[i]  
 gradients = 2 \* xi.T.dot(xi.dot(theta) - yi)  
 theta -= lr \* gradients  
 return theta  
  
# Обычный градиентный спуск (исправленная версия)  
def gradient\_descent(X, y, lr=0.01, epochs=100):  
 theta = np.random.randn(2)  
 X\_b = np.c\_[np.ones((len(X), 1)), X] # Добавляем столбец единиц  
 for epoch in range(epochs):  
 gradients = 2/len(X) \* X\_b.T.dot(X\_b.dot(theta) - y)  
 theta -= lr \* gradients  
 return theta  
  
# Запуск алгоритмов  
theta\_stochastic = stochastic\_gradient\_descent(X, y)  
theta\_standard = gradient\_descent(X, y)  
print(f"\nКоэффициенты SGD: {theta\_stochastic}")  
print(f"Коэффициенты GD: {theta\_standard}")