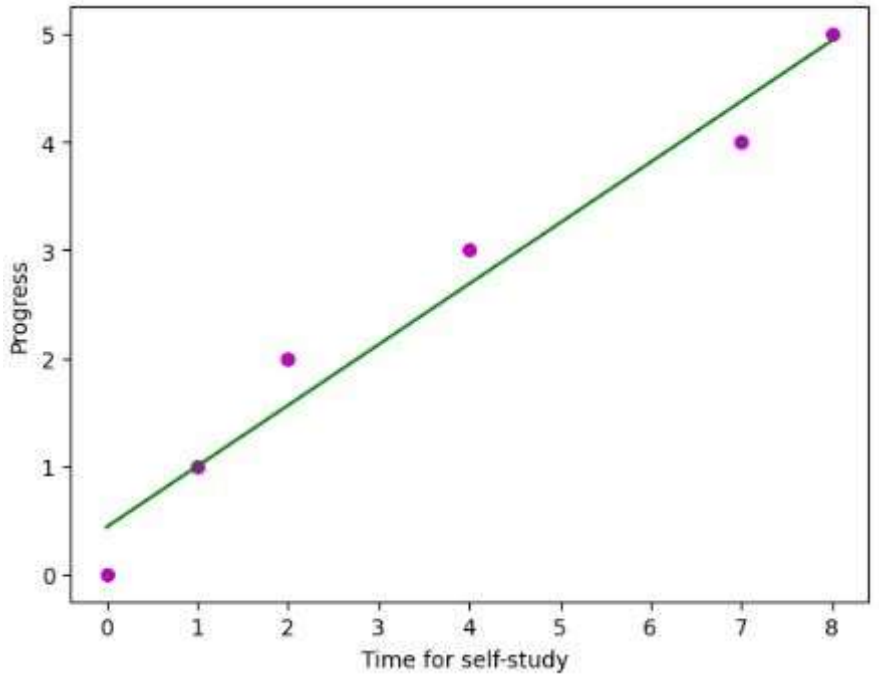
Linear Regression

Простыми словами **линейная регрессия —** это метод, который помогает понять, как *независимая переменная* влияет на *зависимую*. Например, на успеваемость влияет внимательность, дисциплинированность, ответственность, конспектирование, уровень занятости студента, время на самообучение и т.д.

Пример:

- зависимая переменная: оценки (то, что мы хотим предсказать).
- независимая переменная: время на самообучение (то, что мы изменяем).

Если построить график, то можно увидеть, что с увеличением часов учёбы оценки обычно растут. Линейная регрессия помогает нам найти уравнение этой прямой, чтобы, зная количество часов учёбы, мы можем предсказать, какую оценку получит студент.



Линейная регрессия - это процесс моделирования отношений между зависимой переменной с заданным набором независимых переменных. Различают **простую линейную регрессию** (simple linear regression), когда имеется только *одна независимая переменная* и **множественную регрессию** (multiple linear regression), когда *независимых переменных больше*, чем одна.

Задача линейной регрессии - подобрать такие коэффициенты перед X и такой коэффициент смещения (например, перемещающий функцию вверх и вниз на двумерном графике), которые бы максимально точно отражали поведение всех наших наблюдений (data points).

Например, в простой задачи линейной регрессии (один x и один y) форма модели будет иметь вид:

y = $b_0 + b_1 * x + \epsilon$

b_0 - свободный член (пересечение с осью Y)

b_1 - коэффициент наклона (показатель изменения y при изменении x)

x - независимая перменная

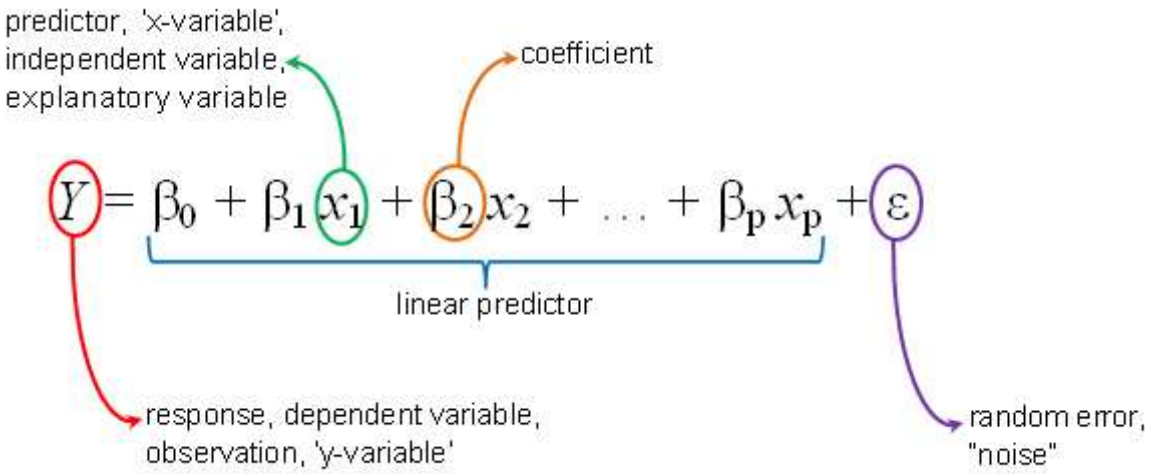
y - зависимая переменная

ϵ — ошибка (разница между наблюдаемыми и предсказанными значениями).

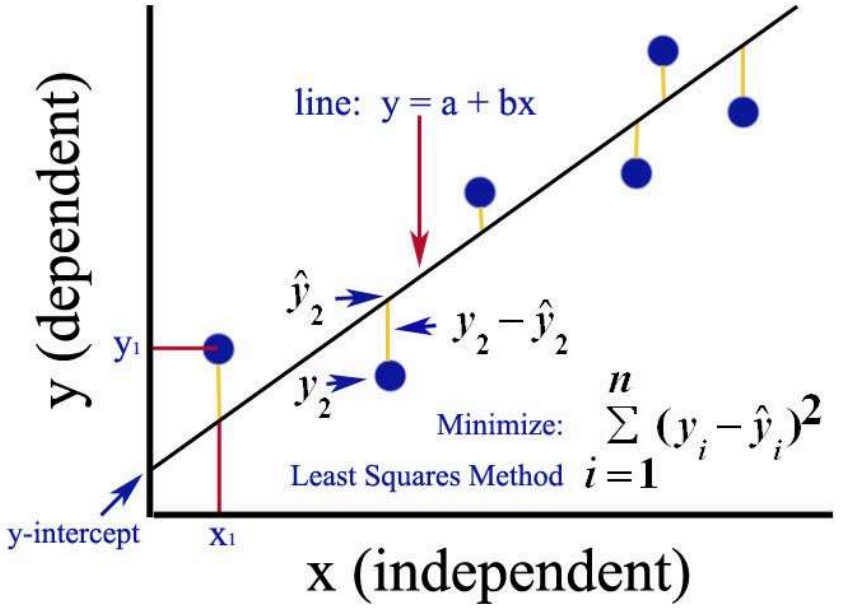
Цель: Найти такие значения b_0 и b_1, чтобы минимизировать сумму квадратов отклонений между наблюдаемыми и предсказанными значениями.

Методы оценки: Наиболее распространённый метод **— метод наименьших квадратов**.

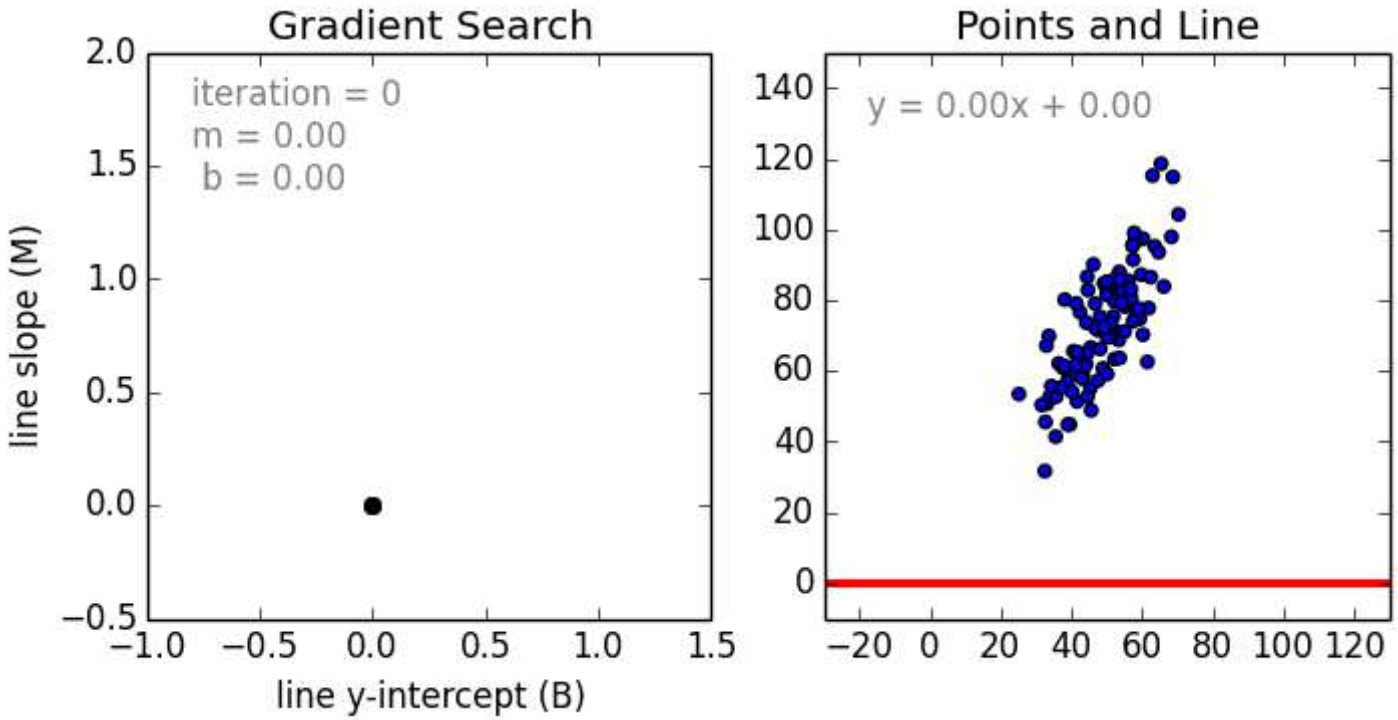
Когда независимых переменных больше, чем одна мы работаем с гиперплоскостью. В общем виде уравнение множественной регрессии с p независимыми переменными имеет вид:



Для минимизации ошибки можно использовать **метод наименьших квадратов**. Его суть заключается в том, чтобы выбрать нужные коэффициенты при минимальной **MSE (среднеквадратичной ошибке)**.



На небольшом наборе данных можно использовать стандартные формулы для нахождения коэффициентов. Однако, если набор данных очень большой, то более рационально воспользоваться **градиентным спуском**, который позволяет оптимизировать значение коэффициентов путем итеративной минимизации ошибки модели в тренировочных данных.



Реализация простой линейной регрессии

- установка необходимых библиотек
- импорт библиотек
- подготовка данных
- разделение данных на обучающую и тестовую выборку
- создание и обучение модели
- сделать предсказание или прогноз
- визуализация результата

In [ ]:
```
pip install numpy pandas scikit-learn matplotlib
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
#  создадим данные для времени на самообучение и оценок
data = {
    'hours_studied': [0, 1, 2, 4, 7, 8],
    'scores': [0, 1, 2, 3, 4, 5]
}
df = pd.DataFrame(data)
```

In [3]:
```
X = df[['hours_studied']]  # независимая переменная
y = df['scores']           # зависимая переменная

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [5]:
```
model = LinearRegression() # построение модели линейной
model.fit(X_train, y_train) # обучение модели
```

Out[5]:
```
LinearRegression()
```

In [6]:
```
predictions = model.predict(X_test)
```

In [7]:
```
plt.scatter(X, y, color='r', label='Данные')
plt.plot(X, model.predict(X), color='g', label='Линейная регрессия')
plt.xlabel('время на самообучение')
plt.ylabel('успеваемость')
plt.title('Простая линейная регрессия')
plt.legend()
plt.show()
```

Модель линейной регрессии может быть задана следующим образом:

y = ax + b

Следовательно, для решения задачи регрессии требуется найти коэффициенты a (коэффициент наклона) и b (точка пересечения линии с осью ординат). Их можно выразить так:

$$a = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

$$b = \bar{y} - ay$$

In [10]:

```
import numpy as np
import matplotlib.pyplot as plt


def estimate_coefficients(x, y):
    # number of observations / количество наблюдений
    n = np.size(x)

    # mean of x and y vector
    mean_x, mean_y = np.mean(x), np.mean(y)

    # calculating cross-deviation and deviation about x
    SS_xy = np.sum(y * x) - n * mean_y * mean_x
    SS_xx = np.sum(x * x) - n * mean_x * mean_x

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = mean_y - b_1 * mean_x

    return b_0, b_1


def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot
    plt.scatter(x, y, color="m", marker="o", s=30)

    # predicted response vector
    y_pred = b[0] + b[1] * x

    # plotting the regression line
    plt.plot(x, y_pred, color="g")

    # putting labels
    plt.xlabel('Time for self-study ')
    plt.ylabel('Progress')

    # function to show plot
    plt.show()


def main():
```

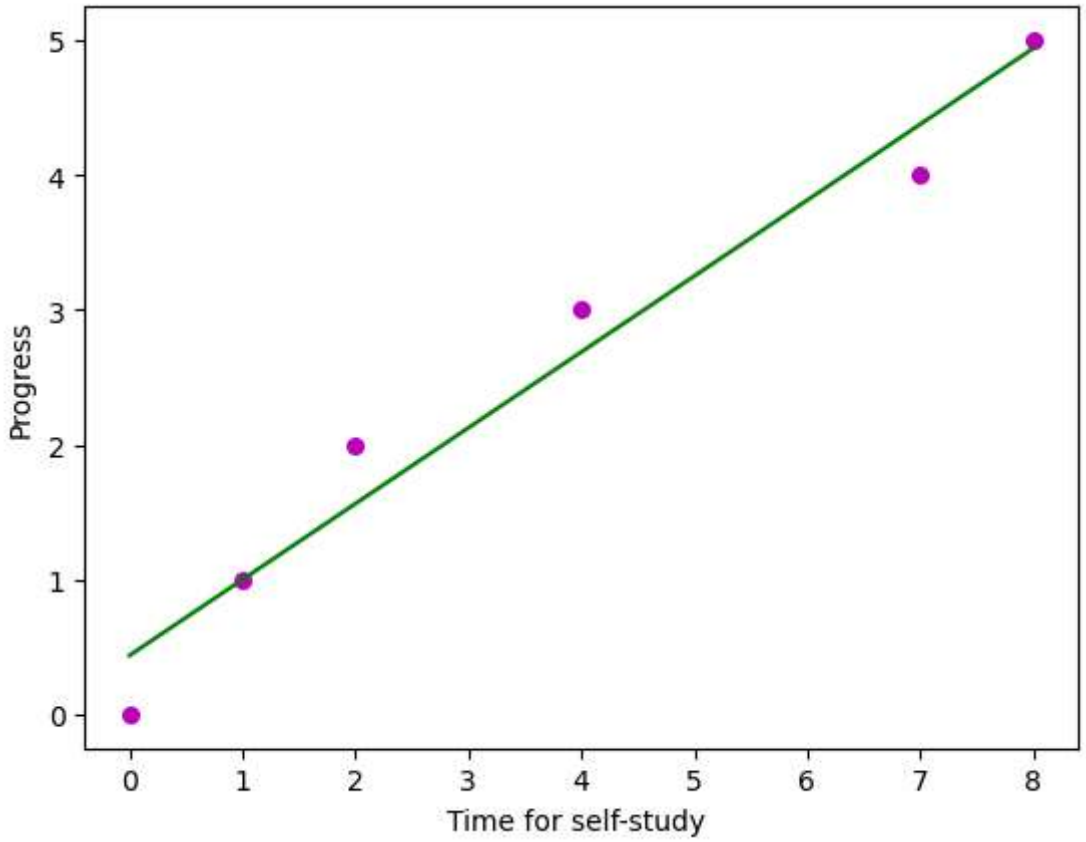
```
    # observations
    x = np.array([0, 1, 2, 4, 7, 8])
    y = np.array([0, 1, 2, 3, 4, 5])

    # estimating coefficients
    b = estimate_coefficients(x, y)
    print("Estimated coefficients:\nb_0 = {} \nb_1 = {}".format(b[0], b[1]))

    # plotting regression line
    plot_regression_line(x, y, b)


if __name__ == "__main__":
    main()
```

```
Estimated coefficients:
b_0 = 0.43750000000000044
b_1 = 0.5624999999999999
```

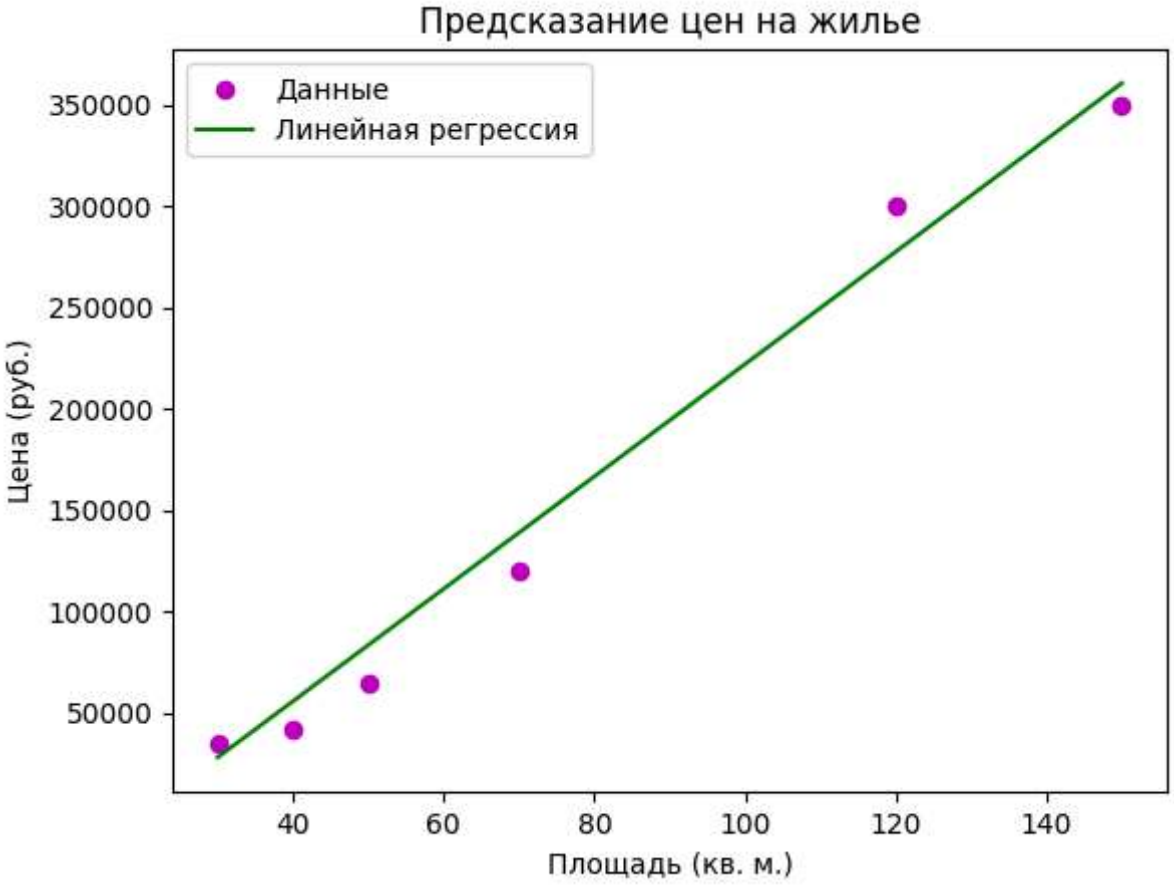


Пример 2. Предсказание цены на жилье в зависимости от площади квартиры

Давайте рассмотрим другую задачу, связанную с линейной регрессией. В этом примере мы попробуем предсказать цены на жилье в зависимости от площади квартиры.

In [11]:
```python
# импорт необходимых библиотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# данные о площади квартир и их цена
data = {
    'area': [30, 40, 50, 70, 120, 150],
    'price': [35000, 42000, 65000, 120000, 300000, 350000]
}
df = pd.DataFrame(data)
# разделим данные на независимую и зависимую переменные
X = df[['area']]  # Независимая переменная (площадь)
y = df['price']   # Зависимая переменная (цена)
# разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
# создание и обучение модели
model = LinearRegression()   # создание модели
model.fit(X_train, y_train)  # обучение модели
# сделаем прогноз на тестовых данных
predictions = model.predict(X_test)
# Estimated coefficients
print(f'Коэффициент наклона (b1): {model.coef_[0]}')
print(f'Свободный член (b0): {model.intercept_}')
# визуализация результатов
plt.scatter(X, y, color='m', label='Данные')
plt.plot(X, model.predict(X), color='g', label='Линейная регрессия')
plt.xlabel('Площадь (кв. м.)')
plt.ylabel('Цена (руб.)')
plt.title('Предсказание цен на жилье')
plt.legend()
plt.show()
```

```
Коэффициент наклона (b1): 2771.386430678466
Свободный член (b0): -55103.24483775807
```

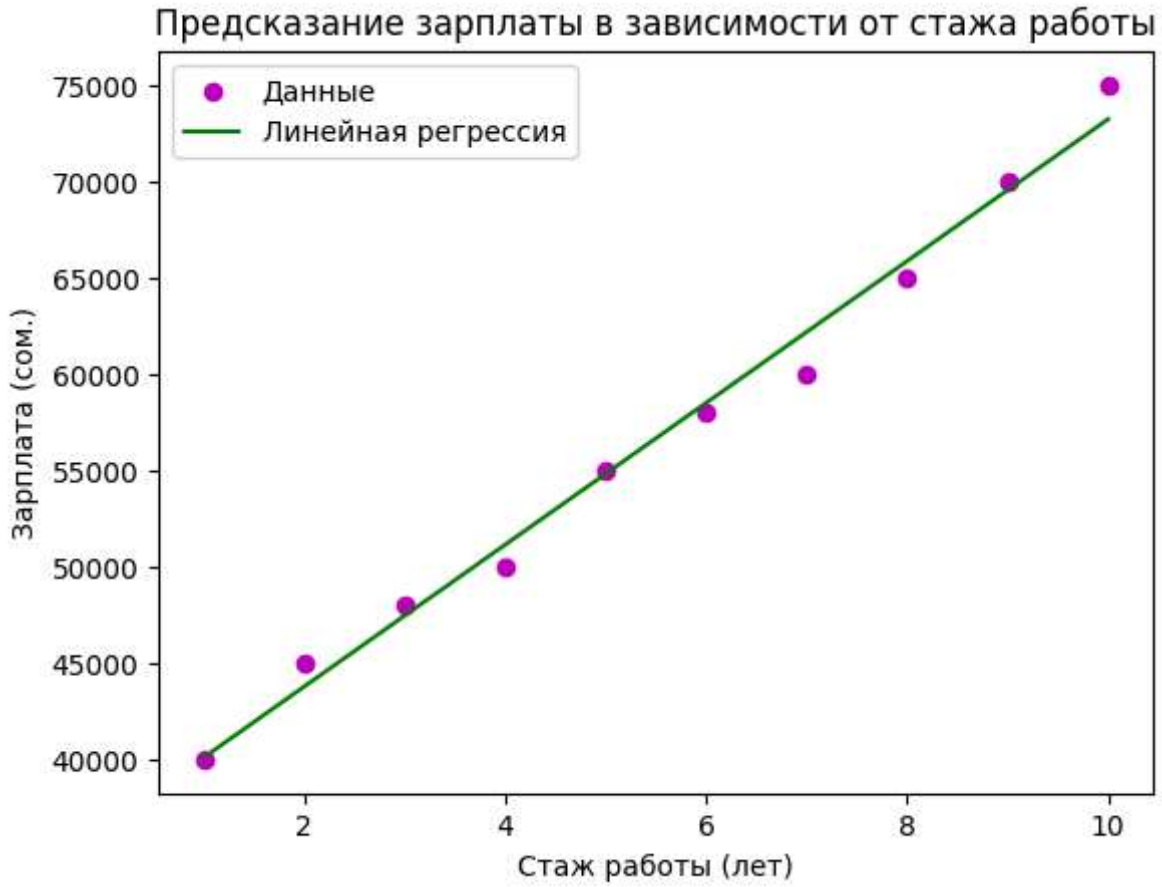


Пример 3. Предсказание зарплаты в зависимости от стажа работы

In [12]:

```python
# импорт необходимых библиотек
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
#  данные о стаже работы сотрудников и их зарплата
data = {
    'experience': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'salary': [40000, 45000, 48000, 50000, 55000, 58000, 60000, 65000, 70000, 75000]
}
df = pd.DataFrame(data)
# разделим данные на независимую и зависимую переменные
X = df[['experience']]  # Независимая переменная (стаж работы)
y = df['salary']        # Зависимая переменная (зарплата)
# разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
# создание и обучение модели
model = LinearRegression()   # создание модели
model.fit(X_train, y_train)  # обучение модели
# сделаем прогноз на тестовых данных
predictions = model.predict(X_test)
# Estimated coefficients
print(f'Коэффициент наклона (b1): {model.coef_[0]}')
print(f'Свободный член (b0): {model.intercept_}')
# визуализация результатов
plt.scatter(X, y, color='m', label='Данные')
plt.plot(X, model.predict(X), color='g', label='Линейная регрессия')
plt.xlabel('Стаж работы (лет)')
plt.ylabel('Зарплата (сом.)')
plt.title('Предсказание зарплаты в зависимости от стажа работы')
plt.legend()
plt.show()
```

```
Коэффициент наклона (b1): 3680.5555555555566
Свободный член (b0): 36416.66666666666
```



Д/з Прочитайте с сайта https://www.dmitrymakarov.ru/data-analysis/eda-04/ (https://www.dmitrymakarov.ru/data-analysis/eda-04/) лекцию и конспектируйте