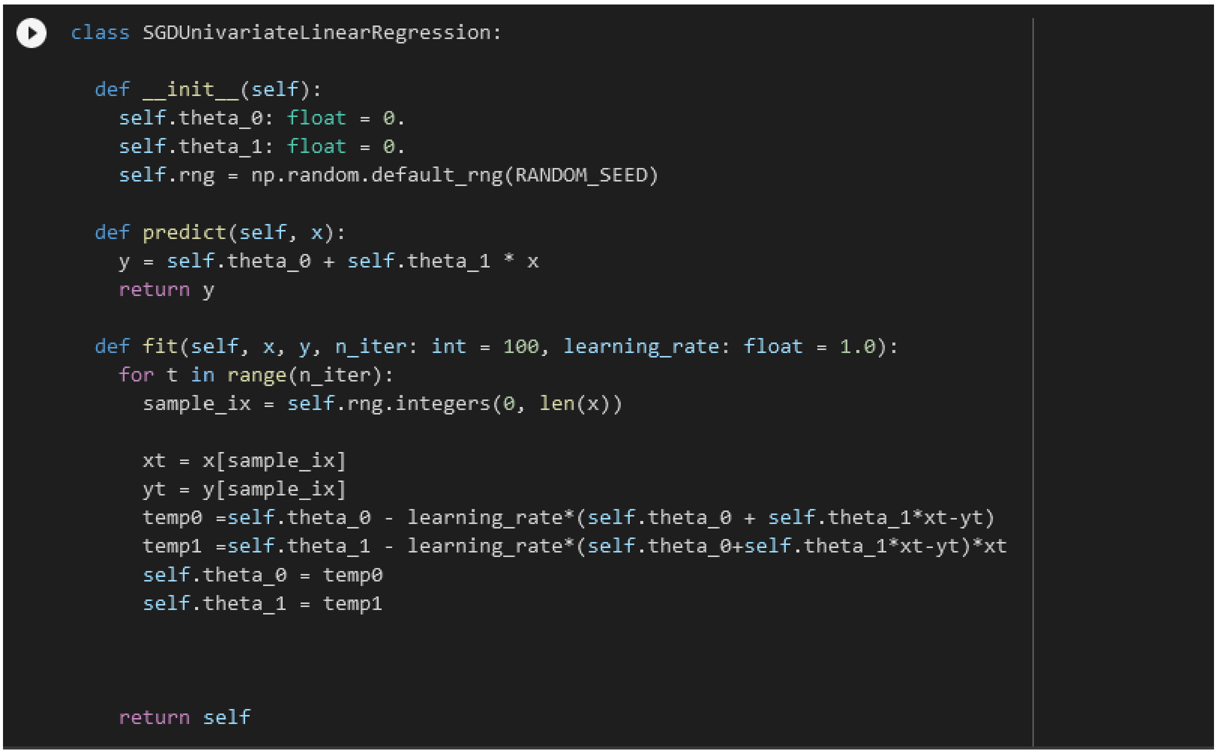
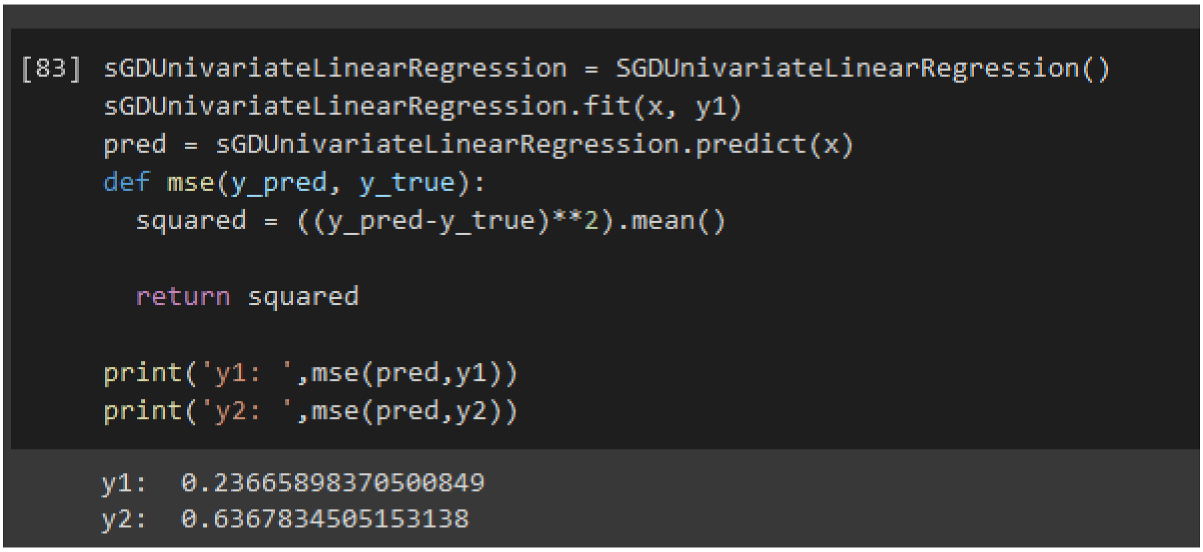
Task 3:

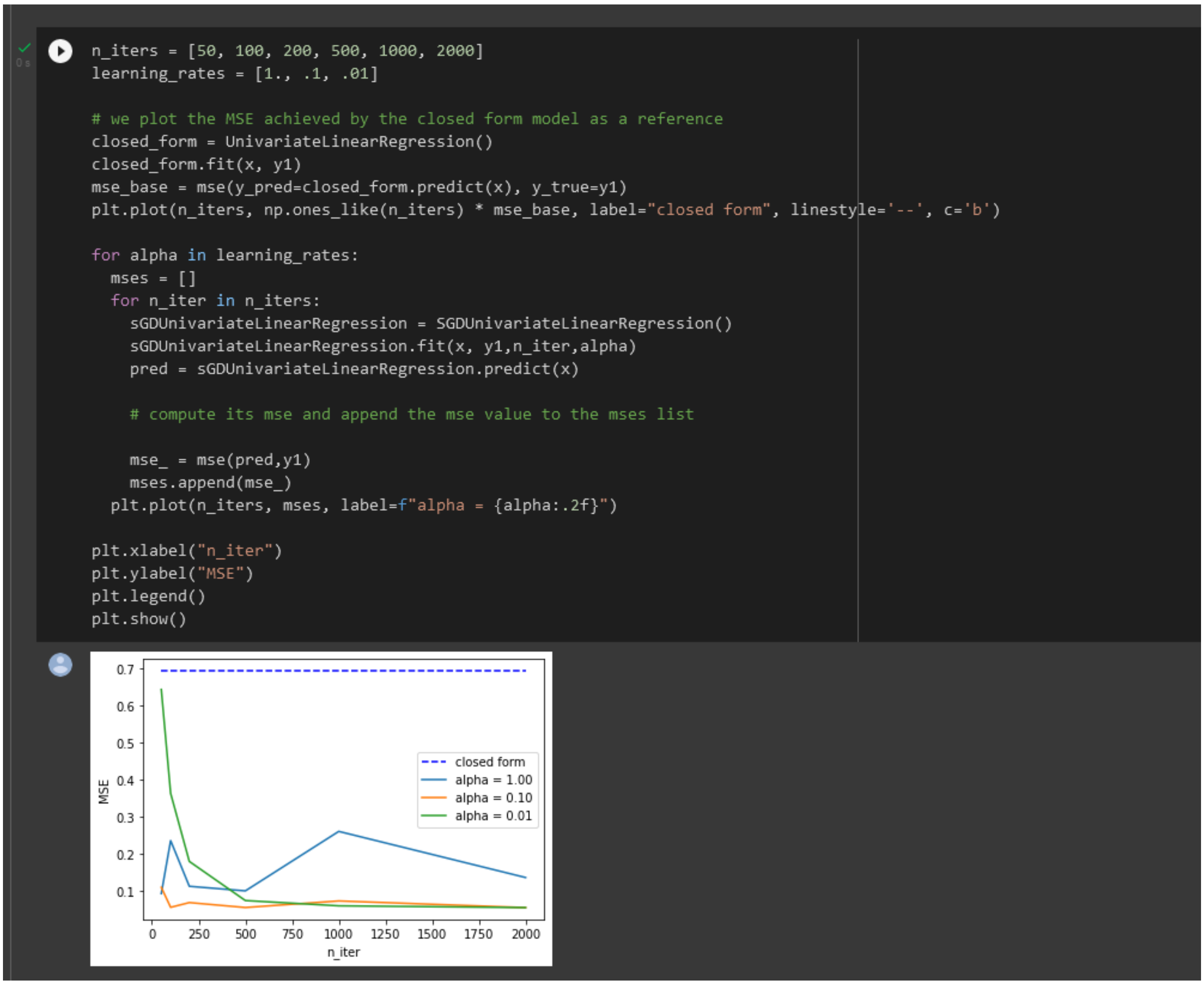


```
class SGDUnivariateLinearRegression:

  def __init__(self):
    self.theta_0: float = 0.
    self.theta_1: float = 0.
    self.rng = np.random.default_rng(RANDOM_SEED)

  def predict(self, x):
    y = self.theta_0 + self.theta_1 * x
    return y

  def fit(self, x, y, n_iter: int = 100, learning_rate: float = 1.0):
    for t in range(n_iter):
      sample_ix = self.rng.integers(0, len(x))

      xt = x[sample_ix]
      yt = y[sample_ix]
      temp0 =self.theta_0 - learning_rate*(self.theta_0 + self.theta_1*xt-yt)
      temp1 =self.theta_1 - learning_rate*(self.theta_0+self.theta_1*xt-yt)*xt
      self.theta_0 = temp0
      self.theta_1 = temp1



    return self
```

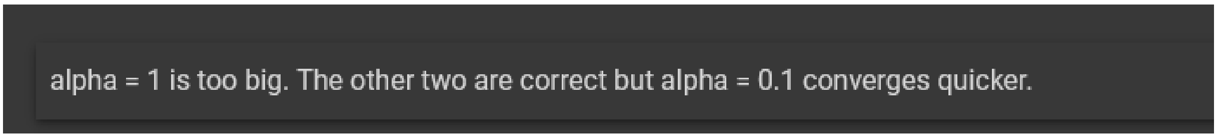


```
[83] sGDUnivariateLinearRegression = SGDUnivariateLinearRegression()
     sGDUnivariateLinearRegression.fit(x, y1)
     pred = sGDUnivariateLinearRegression.predict(x)
     def mse(y_pred, y_true):
       squared = ((y_pred-y_true)**2).mean()

       return squared

     print('y1: ',mse(pred,y1))
     print('y2: ',mse(pred,y2))
```

```
y1:  0.23665898370500849
y2:  0.6367834505153138
```

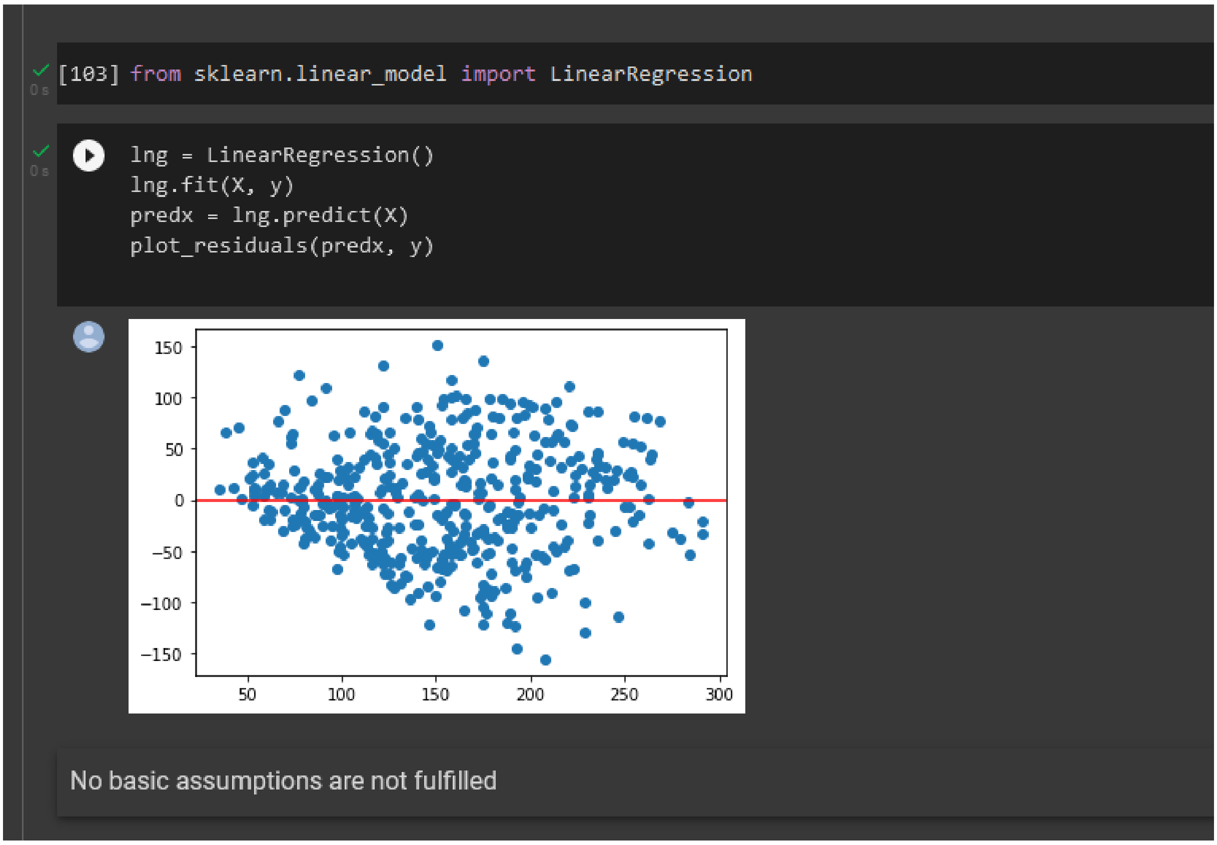


```
n_iters = [50, 100, 200, 500, 1000, 2000]
learning_rates = [1., .1, .01]

# we plot the MSE achieved by the closed form model as a reference
closed_form = UnivariateLinearRegression()
closed_form.fit(x, y1)
mse_base = mse(y_pred=closed_form.predict(x), y_true=y1)
plt.plot(n_iters, np.ones_like(n_iters) * mse_base, label="closed form", linestyle='--', c='b')

for alpha in learning_rates:
  mses = []
  for n_iter in n_iters:
    sGDUnivariateLinearRegression = SGDUnivariateLinearRegression()
    sGDUnivariateLinearRegression.fit(x, y1,n_iter,alpha)
    pred = sGDUnivariateLinearRegression.predict(x)

    # compute its mse and append the mse value to the mses list

    mse_ = mse(pred,y1)
    mses.append(mse_)
  plt.plot(n_iters, mses, label=f"alpha = {alpha:.2f}")

plt.xlabel("n_iter")
plt.ylabel("MSE")
plt.legend()
plt.show()
```



alpha = 1 is too big. The other two are correct but alpha = 0.1 converges quicker.

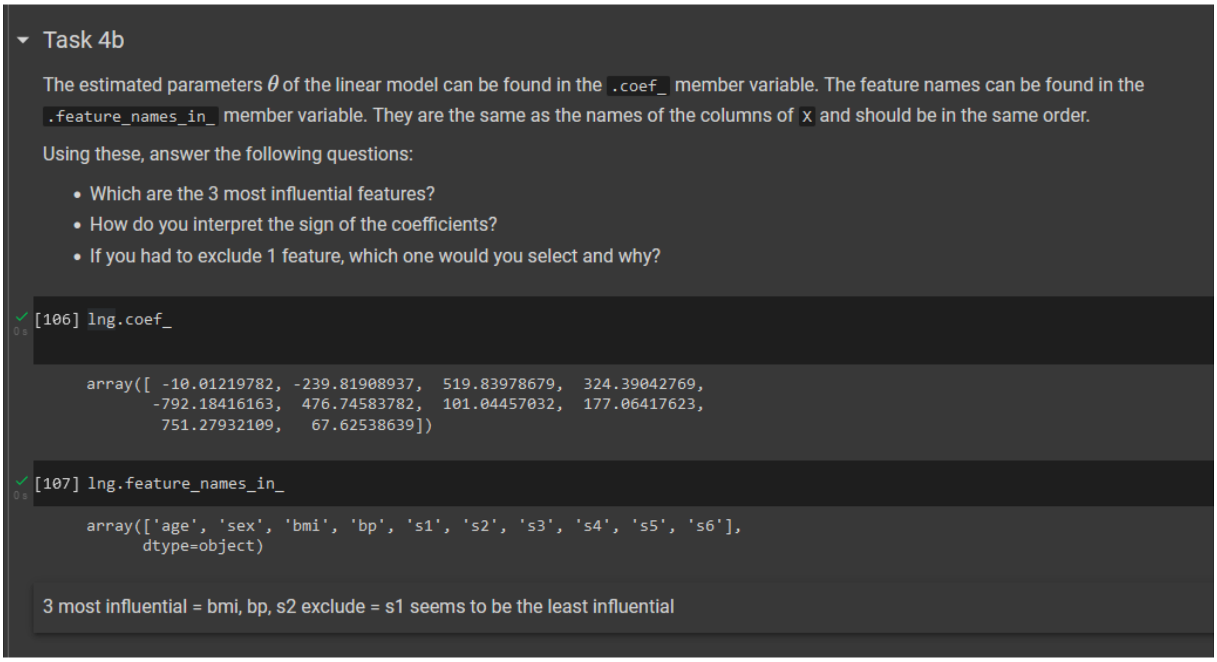
Task 4:



```
[103] from sklearn.linear_model import LinearRegression
```

```
lng = LinearRegression()
lng.fit(X, y)
predx = lng.predict(X)
plot_residuals(predx, y)
```

No basic assumptions are not fulfilled



Task 4b

The estimated parameters θ of the linear model can be found in the `.coef_` member variable. The feature names can be found in the `.feature_names_in_` member variable. They are the same as the names of the columns of `X` and should be in the same order.

Using these, answer the following questions:

- Which are the 3 most influential features?
- How do you interpret the sign of the coefficients?
- If you had to exclude 1 feature, which one would you select and why?

```
[106] lng.coef_

      array([ -10.01219782, -239.81908937,  519.83978679,  324.39042769,
             -792.18416163,  476.74583782,  101.04457032,  177.06417623,
              751.27932109,   67.62538639])
```

```
[107] lng.feature_names_in_

      array(['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6'],
            dtype=object)
```

3 most influential = bmi, bp, s2 exclude = s1 seems to be the least influential