Regression metrics

In this notebook, we will explore some metrics typically applied to linear regression models:

- R2
- Mean squared error (RSS divided by number of samples)
- Ratio of RSS for regression model to sample variance ("RSS for prediction by mean")

using some synthetic data sets.

```
from sklearn import datasets
from sklearn import metrics
from sklearn import preprocessing
from sklearn.linear_model import LinearRegression

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

Generate synthetic data

We will generate four sets of synthetic data for a simple linear regression.

Each dataset will be generated using the `make_regression` function in `sklearn`'s `datasets` module. This will:

- generate a random regression coefficient, w_1,
- generate `n_samples` points on the line defined by that coefficient (i.e. generate random x and then compute y using the equation for the linear model),
- and then add Gaussian noise with standard deviation defined by the `noise` argument to each of the `n_samples` points.

We will also scale all the "features" to the $[-1, 1]$ range using `sklearn`'s `MaxAbsScaler`, so that we can make reasonable comparisons between the datasets.

The sets `hivar1` and `lovar1` will be identical to one another with respect to the number of samples and regression coefficents, but the `hivar1` set will have 5x the noise of the `lovar1` set.

Similarly, the sets `hivar2` and `lovar2` will be identical to one another with respect to the number of samples and regression coefficents, but the `hivar2` set will have 5 times the noise of the `lovar2` set.

```
X_hivar1, y_hivar1 = datasets.make_regression(n_samples=300, n_features=1, noise=20,
    random_state=4)
X_hivar1 = preprocessing.MaxAbsScaler().fit_transform(X_hivar1)

X_lovar1, y_lovar1 = datasets.make_regression(n_samples=300, n_features=1, noise=4,
    random_state=4)
X_lovar1 = preprocessing.MaxAbsScaler().fit_transform(X_lovar1)

X_hivar2, y_hivar2 = datasets.make_regression(n_samples=150, n_features=1, noise=50,
    random_state=9)
X_hivar2 = preprocessing.MaxAbsScaler().fit_transform(X_hivar2)
```

```
X_lovar2, y_lovar2 = datasets.make_regression(n_samples=150, n_features=1, noise=10,
    random_state=9)
X_lovar2 = preprocessing.MaxAbsScaler().fit_transform(X_lovar2)
```

Fit a linear regression

Next, we will fit a linear regression to each data set:

```
regr_hivar1 = LinearRegression().fit(X_hivar1, y_hivar1)
regr_lovar1 = LinearRegression().fit(X_lovar1, y_lovar1)
regr_hivar2 = LinearRegression().fit(X_hivar2, y_hivar2)
regr_lovar2 = LinearRegression().fit(X_lovar2, y_lovar2)
```

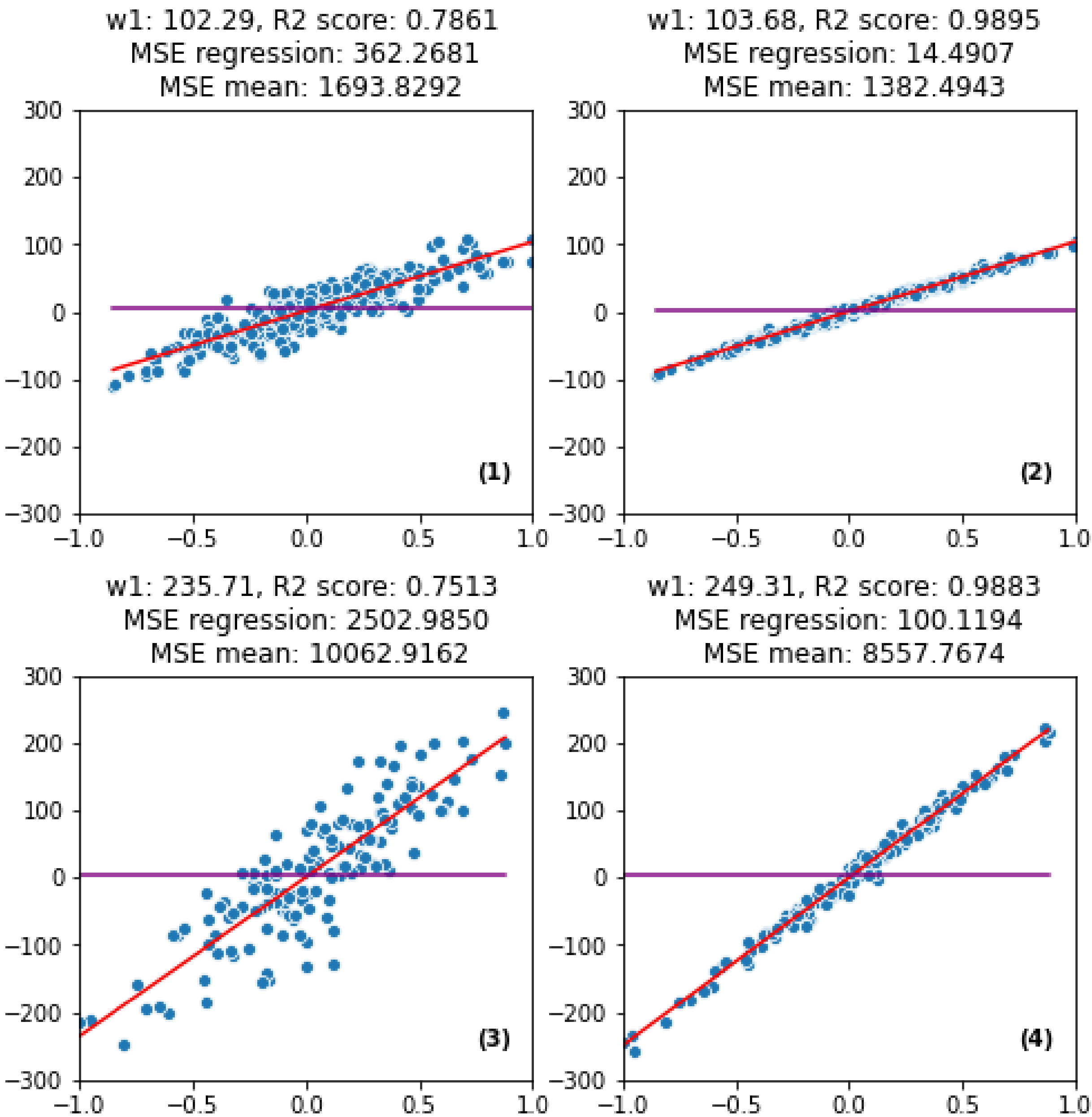
Visualize data and regression line

Finally, for each dataset:

- we plot the data points and the fitted linear regression line
- we print the coefficient w_1 on each plot
- we print the R2 value on each plot
- we compute the MSE of the regression, and print it on each plot
- we compute the "MSE of prediction by mean", and print it on each plot

```
fig = plt.figure()
fig.set_size_inches(8, 8)
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)
plt.subplots_adjust(hspace=0.4)

sns.scatterplot(x=X_hivar1.squeeze(), y=y_hivar1, ax=ax1);
sns.lineplot(x=X_hivar1.squeeze(), y=regr_hivar1.predict(X_hivar1), color='red', ax=ax1);
sns.lineplot(x=X_hivar1.squeeze(), y=np.mean(y_hivar1), color='purple', ax=ax1);
ax1.title.set_text('w1: %s, R2 score: %s \n MSE regression: %s \n MSE mean: %s' %
          (
           '{0:.2f}'.format(regr_hivar1.coef_[0]),
           '{0:.4f}'.format(metrics.r2_score(y_hivar1, regr_hivar1.predict(X_hivar1))),
           '{0:.4f}'.format(np.mean((regr_hivar1.predict(X_hivar1)-y_hivar1)**2)),
           '{0:.4f}'.format(np.mean(( np.mean(y_hivar1)-y_hivar1)**2))
          ));
ax1.text(0.75, -250, "(1)", size='medium', color='black', weight='semibold');
ax1.set_ylim(-300, 300);
ax1.set_xlim(-1, 1);

sns.scatterplot(x=X_lovar1.squeeze(), y=y_lovar1, ax=ax2);
sns.lineplot(x=X_lovar1.squeeze(), y=regr_lovar1.predict(X_lovar1), color='red', ax=ax2);
sns.lineplot(x=X_lovar1.squeeze(), y=np.mean(y_lovar1), color='purple', ax=ax2);
ax2.title.set_text('w1: %s, R2 score: %s \n MSE regression: %s \n MSE mean: %s' %
          (
           '{0:.2f}'.format(regr_lovar1.coef_[0]),
           '{0:.4f}'.format(metrics.r2_score(y_lovar1, regr_lovar1.predict(X_lovar1))),
           '{0:.4f}'.format(np.mean((regr_lovar1.predict(X_lovar1)-y_lovar1)**2)),
           '{0:.4f}'.format(np.mean(( np.mean(y_lovar1)-y_lovar1)**2))
```

```
        ));
ax2.text(0.75, -250, "(2)", size='medium', color='black', weight='semibold');
ax2.set_ylim(-300, 300);
ax2.set_xlim(-1, 1);

sns.scatterplot(x=X_hivar2.squeeze(), y=y_hivar2, ax=ax3);
sns.lineplot(x=X_hivar2.squeeze(), y=regr_hivar2.predict(X_hivar2), color='red', ax=ax3);
sns.lineplot(x=X_hivar2.squeeze(), y=np.mean(y_hivar2), color='purple', ax=ax3);
ax3.title.set_text('w1: %s, R2 score: %s \n MSE regression: %s \n MSE mean: %s' %
        (
         '{0:.2f}'.format(regr_hivar2.coef_[0]),
         '{0:.4f}'.format(metrics.r2_score(y_hivar2, regr_hivar2.predict(X_hivar2))),
         '{0:.4f}'.format(np.mean((regr_hivar2.predict(X_hivar2)-y_hivar2)**2)),
         '{0:.4f}'.format(np.mean(( np.mean(y_hivar2)-y_hivar2)**2))
        ));
ax3.text(0.75, -250, "(3)", size='medium', color='black', weight='semibold');
ax3.set_ylim(-300, 300);
ax3.set_xlim(-1, 1);

sns.scatterplot(x=X_lovar2.squeeze(), y=y_lovar2, ax=ax4);
sns.lineplot(x=X_lovar2.squeeze(), y=regr_lovar2.predict(X_lovar2), color='red', ax=ax4);
sns.lineplot(x=X_lovar2.squeeze(), y=np.mean(y_lovar2), color='purple', ax=ax4);
ax4.title.set_text('w1: %s, R2 score: %s \n MSE regression: %s \n MSE mean: %s' %
        (
         '{0:.2f}'.format(regr_lovar2.coef_[0]),
         '{0:.4f}'.format(metrics.r2_score(y_lovar2, regr_lovar2.predict(X_lovar2))),
         '{0:.4f}'.format(np.mean((regr_lovar2.predict(X_lovar2)-y_lovar2)**2)),
         '{0:.4f}'.format(np.mean(( np.mean(y_lovar2)-y_lovar2)**2))
        ));
ax4.text(0.75, -250, "(4)", size='medium', color='black', weight='semibold');
ax4.set_ylim(-300, 300);
ax4.set_xlim(-1, 1);
```



Interpret results

Based on the figures above, we can make the following statements:

From w_1, and visually from the slope of the regression line:

- For **(1)**, **(2)**: an increase in x of 1 is, on average, associated with an increase in y of about 100.
- For **(3)**, **(4)**: an increase in x of 1 is, on average, associated with an increase in y of about 240.

From the R2 score, and visually from the variance around the regression line:

- For **(1)**, **(3)**: about 75% of the variance in y is explained by the regression on x.
- For **(2)**, **(4)**: about 99% of the variance in y is explained by the regression on x.

We also observe:

- The MSE of the regression line is equivalent to the variance of the noise we added around the regression line. (Take the square of the `noise` argument we used, which was the standard deviation of the noise.)
- The greater the slope of the regression line, the more error is associated with prediction by mean. Prediction by mean is the same thing as prediction by a line with intercept $w_0 = \overline{y}$ and slope $w_1 = 0$ (purple line in the figures above). The greater the true w_1, the more "wrong" the $w_1 = 0$ prediction is.
- The ratio of MSE of the regression line to MSE of prediction by mean, is $1 - R2$.