Special Topics
REGRESSION

PHYS 453

Dr Daugherity

ACU
ABILENE CHRISTIAN UNIVERSITY

Regression

Instead of classifying into categories, our job is now to predict a real-valued number based on training data

Basically all of the techniques we've learned can be modified to work for regression

(PS – Remember that LogisticRegression is actually a classifier…)

Regression

Examples:

- predict the price of a house in Boston
- predict how much money a movie will make
- guess the value of a stock
- Diabetes dataset: predict measure of disease based on patient data

Can a data set be unethical? For a wild ride look at some controversy regarding the Boston housing set:

- https://medium.com/@docintangible/racist-data-destruction-113e3eff54a8
- https://fairlearn.org/main/user_guide/datasets/boston_housing_data.html

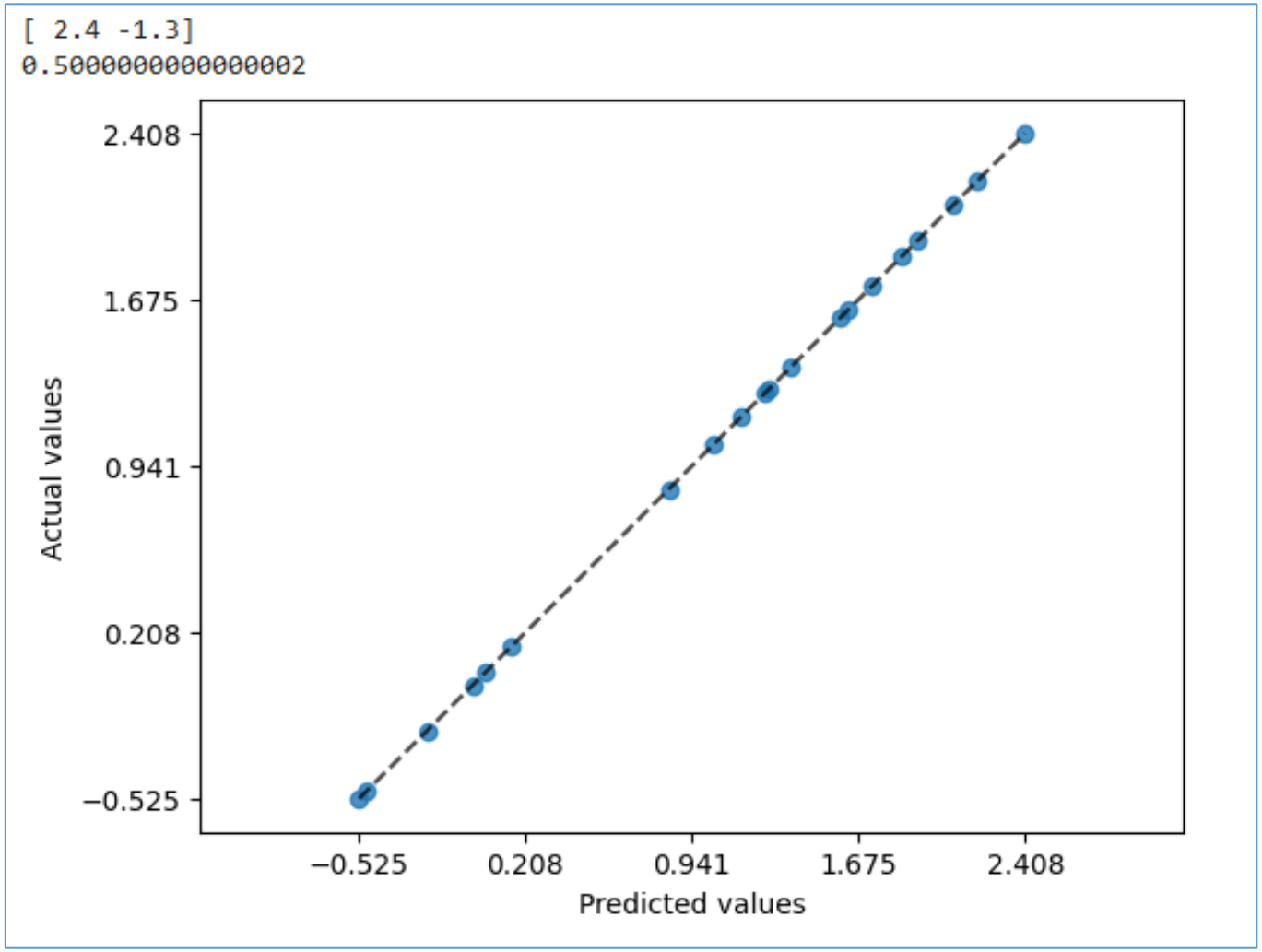
Coding Regression

- Essentially the same as classification in sklearn
- Still good to use test/train split
- Still use X and y, but y is now “target” values

```
NUM_FEAT = 2
NUM_SAMP = 20
X = np.random.rand(NUM_SAMP, NUM_FEAT)
coef = np.array([2.4, -1.3])
offset = 0.5
y = np.dot(X, coef) + offset

reg = linear_model.LinearRegression()
reg.fit(X, y)
y_pred = reg.predict(X)
PredictionErrorDisplay.from_predictions(y,
 y_pred, kind="actual_vs_predicted")

print(reg.coef_)
print(reg.intercept_)
```

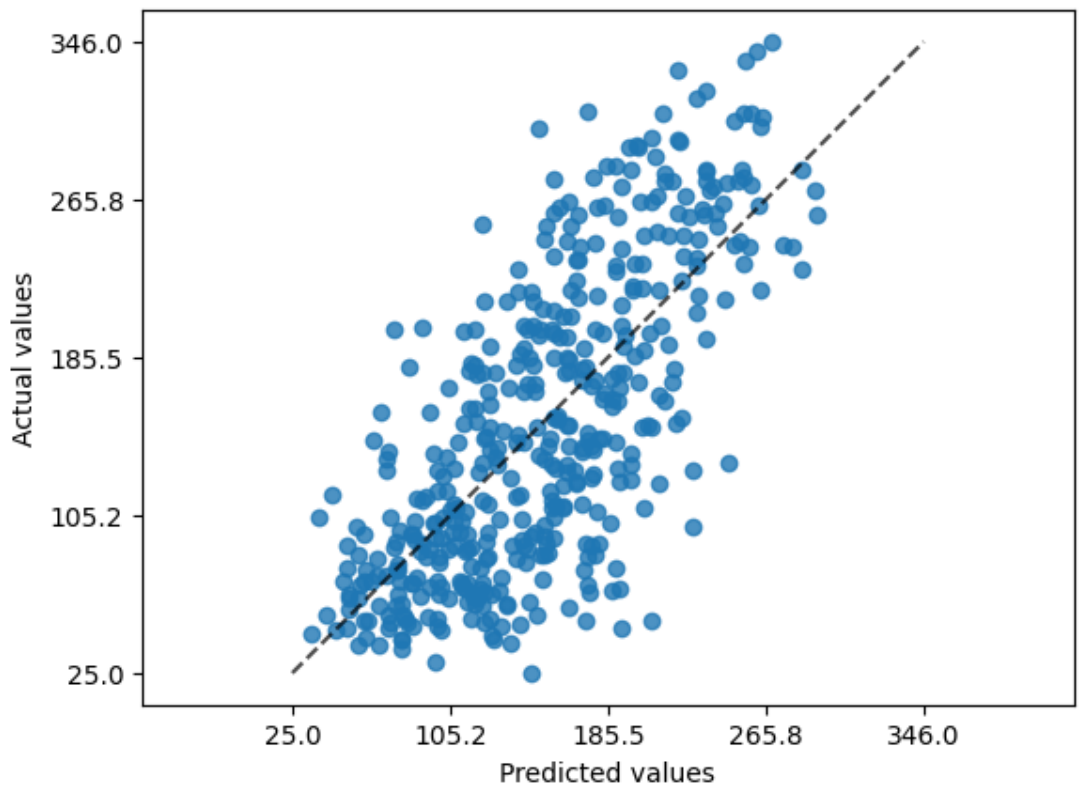


```
1 from sklearn import datasets
```

```
1 X,y = datasets.load_diabetes(return_X_y=True)
2 print(X.shape)
```

```
(442, 10)
```

```
1 reg = linear_model.LinearRegression()
2 reg.fit(X, y)
3 y_pred = reg.predict(X)
4 PredictionErrorDisplay.from_predictions(y, y_pred, kind="actual_vs_predicted")
5
```



Regression Scores

See the User's Guide on Regression Metrics:
https://scikit-learn.org/stable/modules/model_evaluation.html#regression-metrics

For classifiers when we call the score function we get % accuracy. Remember there are fancy measures like precision, recall, f1-score, etc.

For regressors the default is called the **coefficient of determination aka R^2 score**.

- Perfect score is 1.0
- Always predicting the mean y (regardless of features) is 0
- Score can be negative since prediction can be arbitrarily bad

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Regression Scores

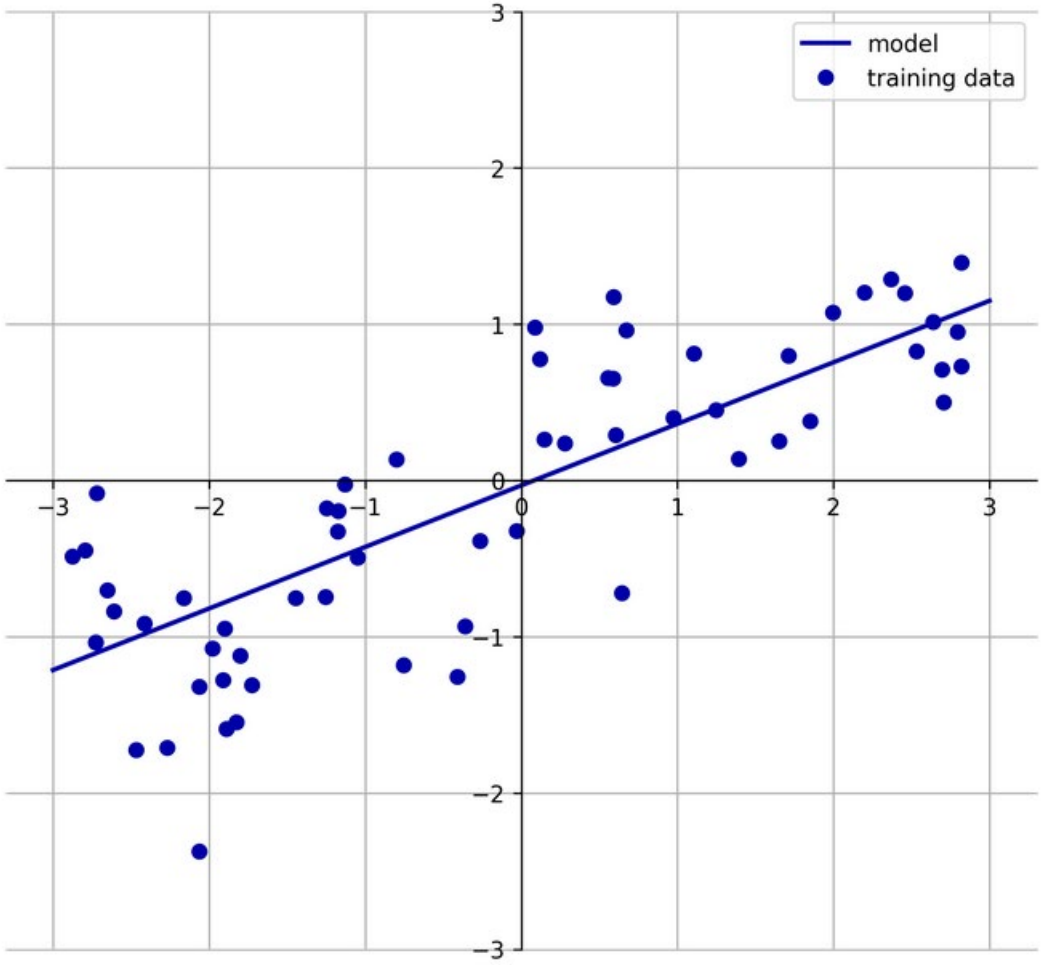
Other choices available:
https://scikit-learn.org/stable/modules/classes.html#regression-metrics

metrics.explained_variance_score(y_true, ...)	Explained variance regression score function.
metrics.max_error(y_true, y_pred)	The max_error metric calculates the maximum residual error.
metrics.mean_absolute_error(y_true, y_pred, *)	Mean absolute error regression loss.
metrics.mean_squared_error(y_true, y_pred, *)	Mean squared error regression loss.
metrics.mean_squared_log_error(y_true, y_pred, *)	Mean squared logarithmic error regression loss.
metrics.median_absolute_error(y_true, y_pred, *)	Median absolute error regression loss.
metrics.mean_absolute_percentage_error(...)	Mean absolute percentage error (MAPE) regression loss.
metrics.r2_score(y_true, y_pred, *[, ...])	R^2 (coefficient of determination) regression score function.
metrics.root_mean_squared_log_error(y_true, ...)	Root mean squared logarithmic error regression loss.
metrics.root_mean_squared_error(y_true, ...)	Root mean squared error regression loss.
metrics.mean_poisson_deviance(y_true, y_pred, *)	Mean Poisson deviance regression loss.
metrics.mean_gamma_deviance(y_true, y_pred, *)	Mean Gamma deviance regression loss.
metrics.mean_tweedie_deviance(y_true, y_pred, *)	Mean Tweedie deviance regression loss.
metrics.d2_tweedie_score(y_true, y_pred, *)	D^2 regression score function, fraction of Tweedie deviance explained.
metrics.mean_pinball_loss(y_true, y_pred, *)	Pinball loss for quantile regression.
metrics.d2_pinball_score(y_true, y_pred, *)	D^2 regression score function, fraction of pinball loss explained.
metrics.d2_absolute_error_score(y_true, ...)	D^2 regression score function, fraction of absolute error explained.

Linear Regression

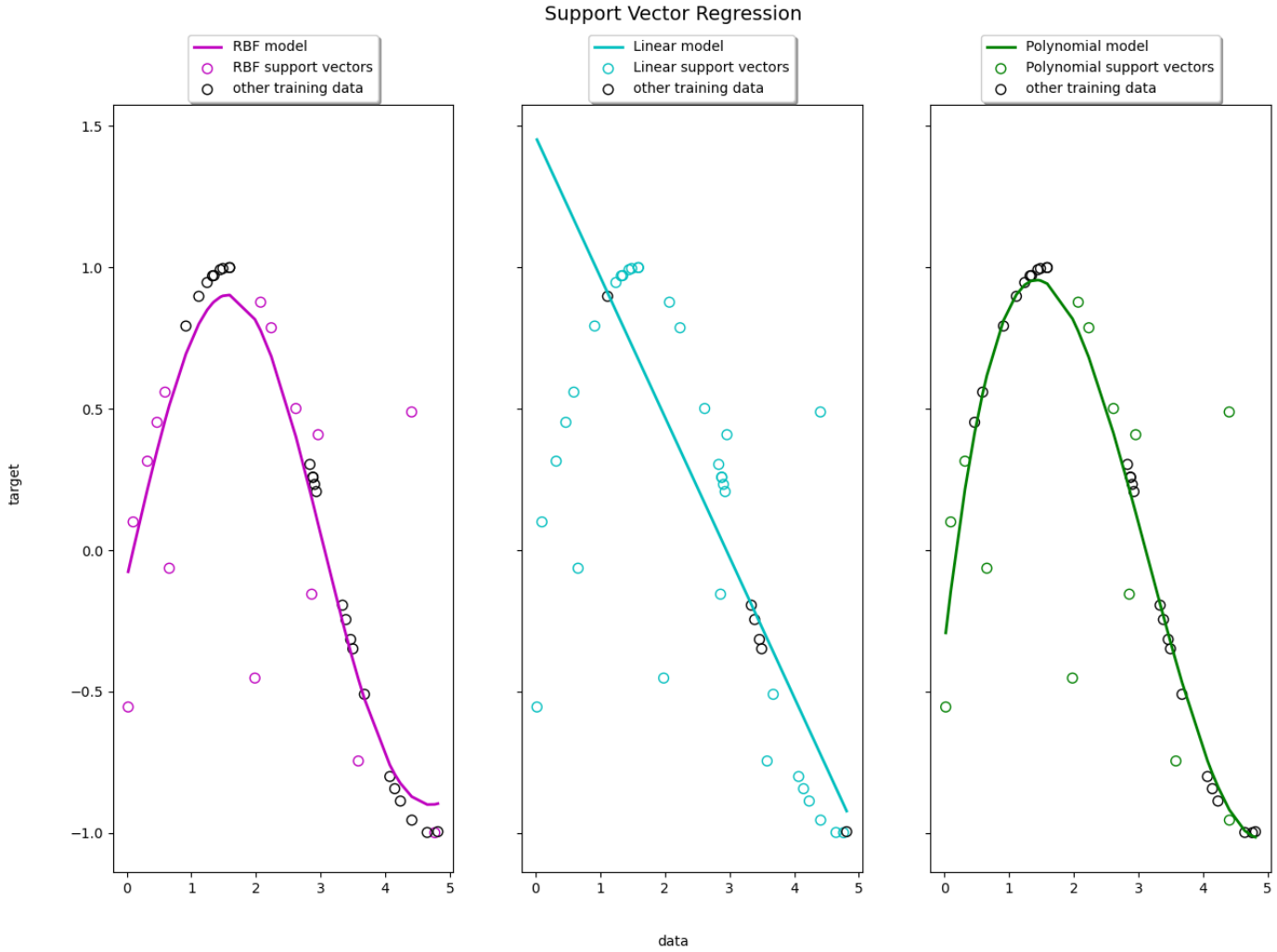
- Literally just a fit to the data
- Usually implemented as finding the parameters (slopes and intercepts) that minimize a least-squares error
- Variations on this are the ridge and lasso regressions, idea is just to add another penalty to error function
- Coefficients relate to feature importance! (But be careful, we really mean than features are important *to this model*, not necessarily the dataset itself)

https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb



Support Vector Regression

- In classification the SVC ignores most data points and maximizes the margin between the “support vectors” (points closest to boundary)
- In regression the SVR ignores points within ϵ of the line, so it chases outliers
- Can also do non-linear kernels!

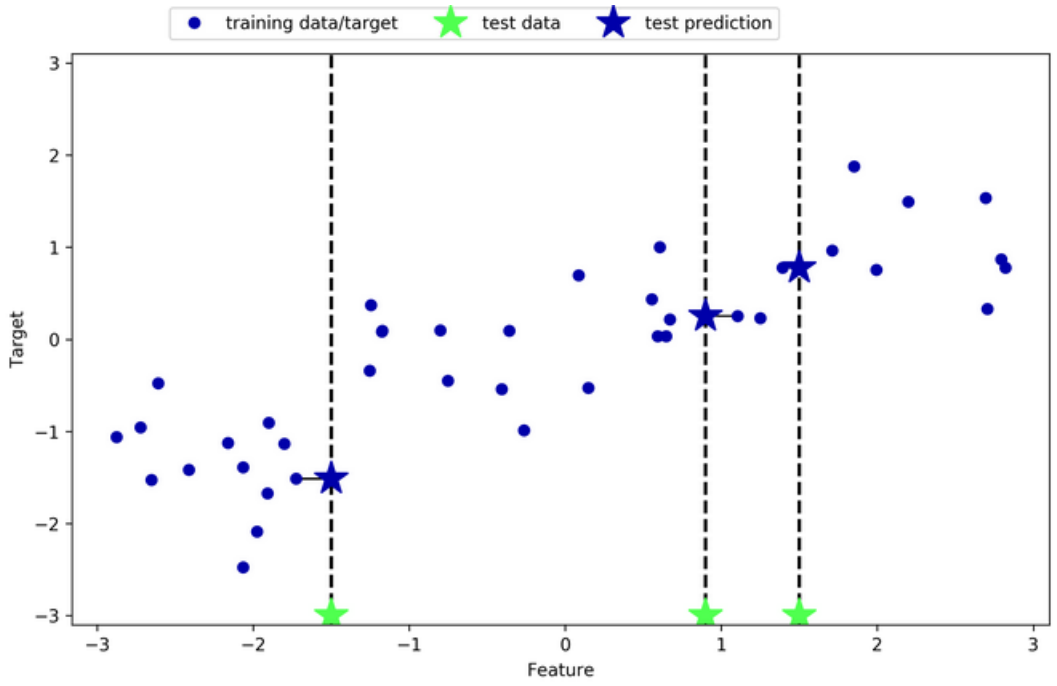


https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html#sphx-glr-auto-examples-svm-plot-svm-regression-py

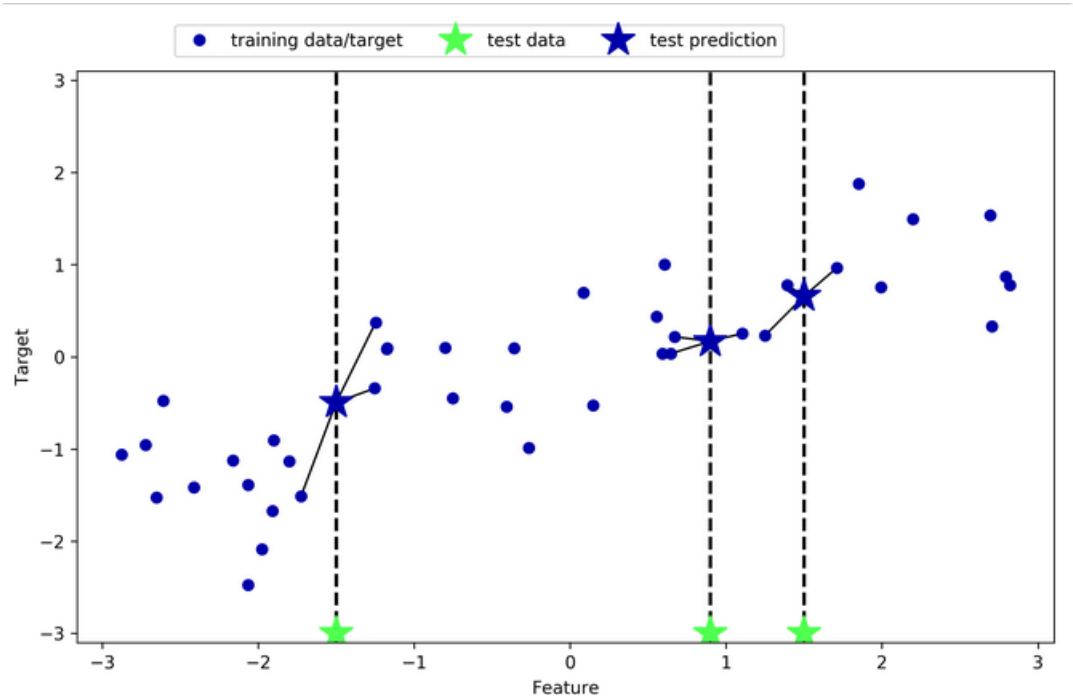
KNN Regression

- Simple idea: predict average of k neighbors!

https://github.com/amueller/introduction_to_ml_with_python/blob/master/02-supervised-learning.ipynb

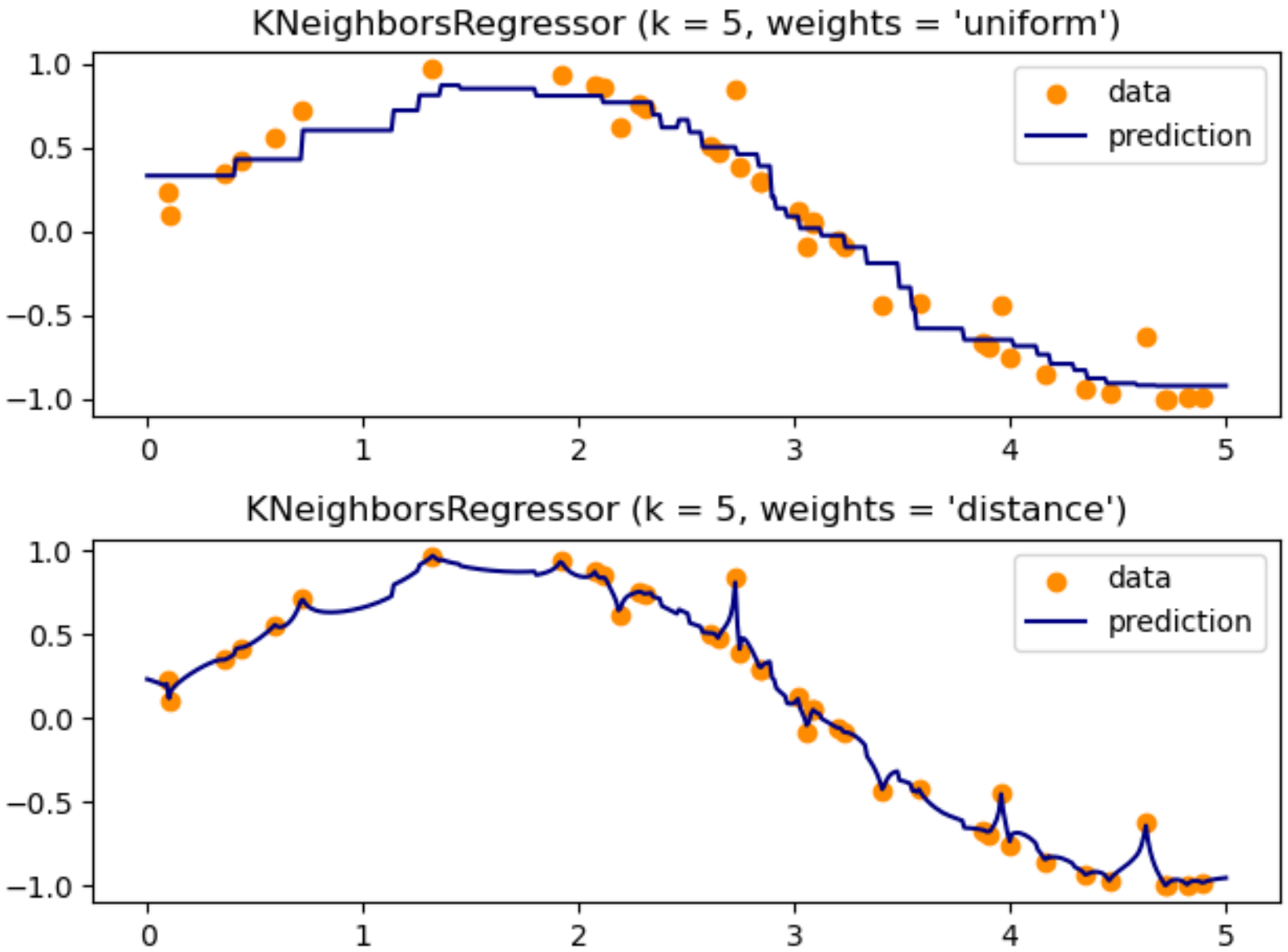


k=1



k=3

https://scikit-learn.org/stable/auto_examples/neighbors/plot_regression.html#sphx-glr-auto-examples-neighbors-plot-regression-py

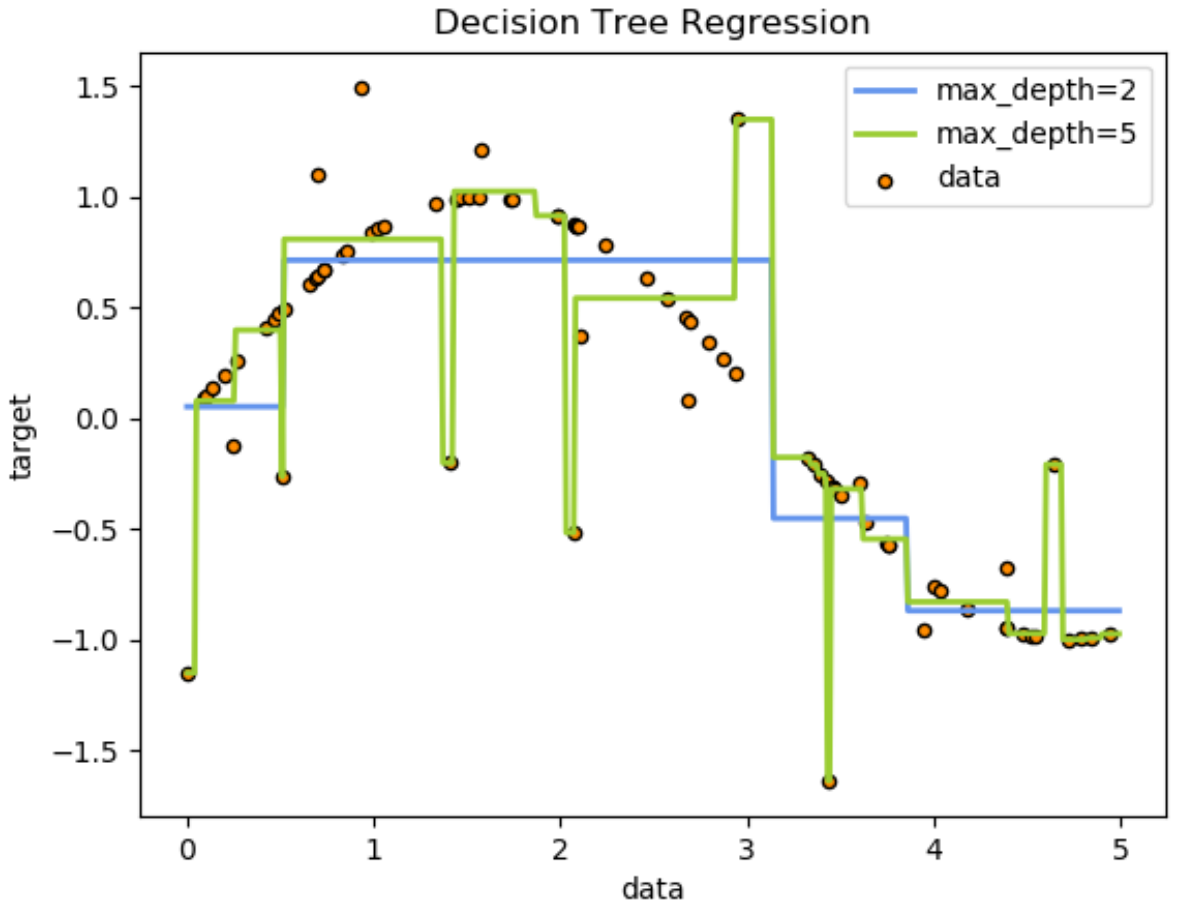


How sensitive should we be to noise and outliers?

Decision Tree Regressor

- Similar to KNN, bad at extrapolation!
- Tends to stair-step or overfit

http://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

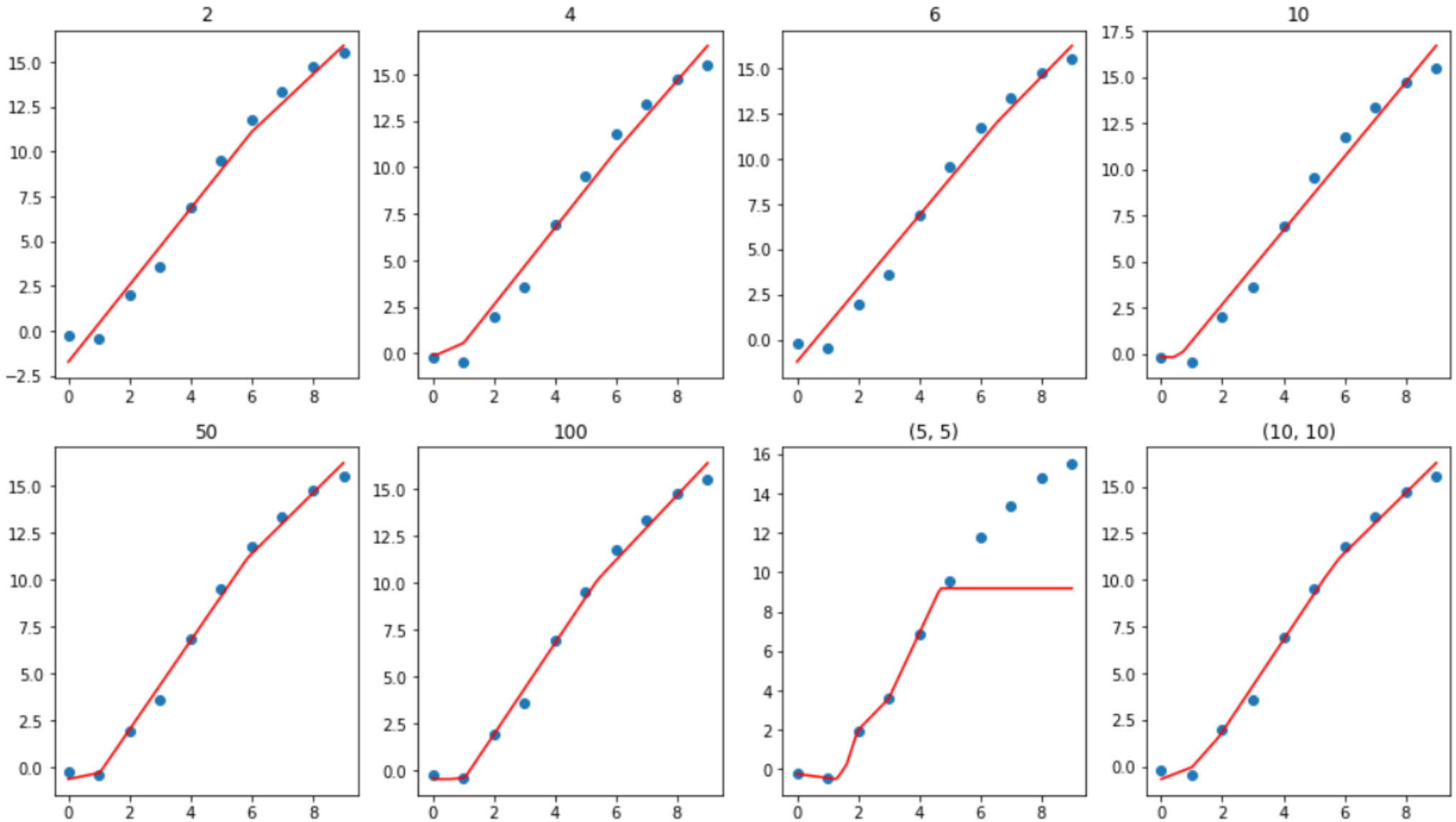


https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_multioutput_face_completion.html#sphx-glr-auto-examples-miscellaneous-plot-multioutput-face-completion-py

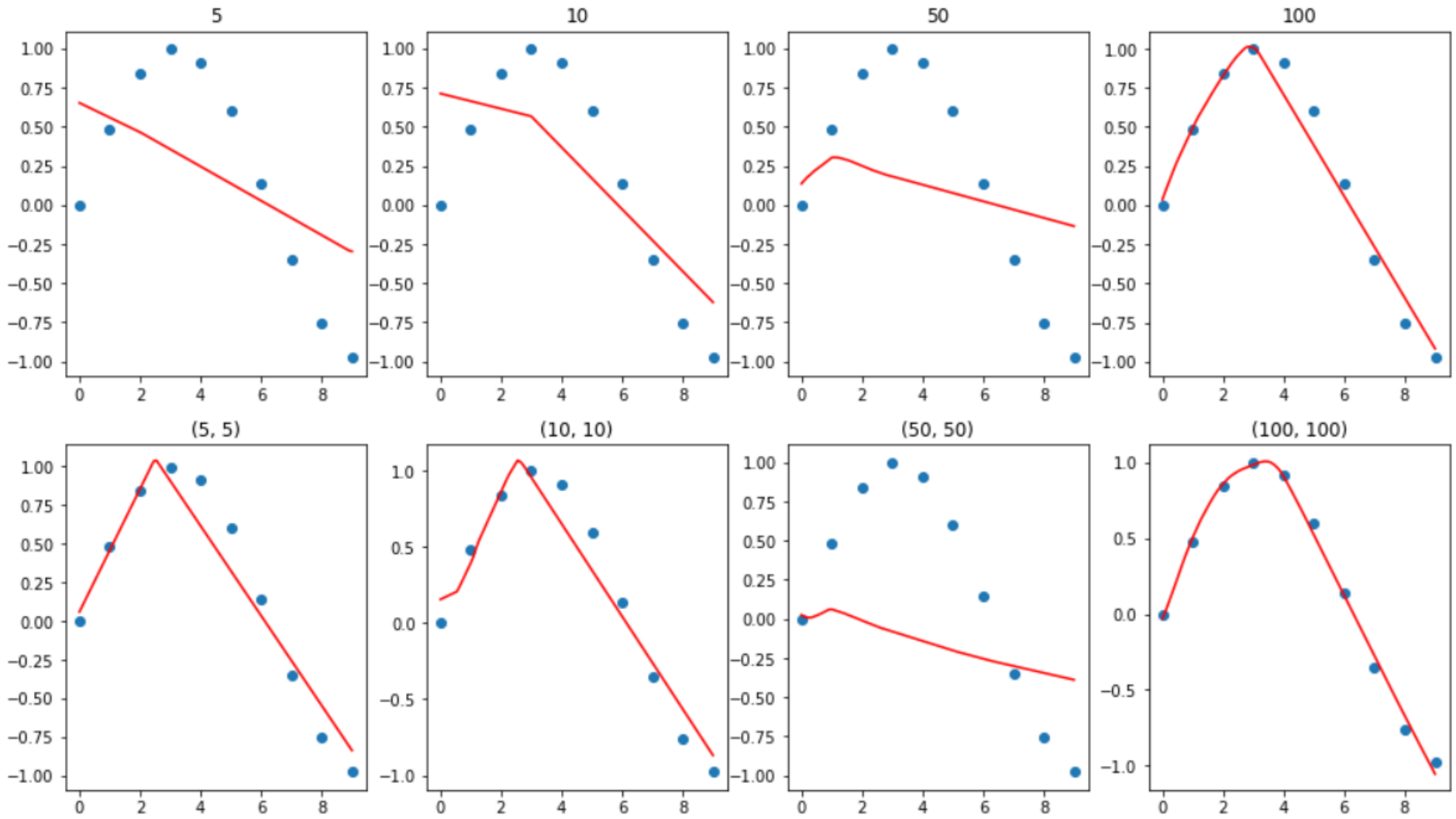
Given the top half of a face as features, predict each pixel in the bottom face as targets

Neural Network Regressor

- No activation function on output layer
- Adjust weights to minimize MSE
- Hidden layers affect complexity of curve



Fit a NN to the 10 blue data points. Titles show hidden layers.

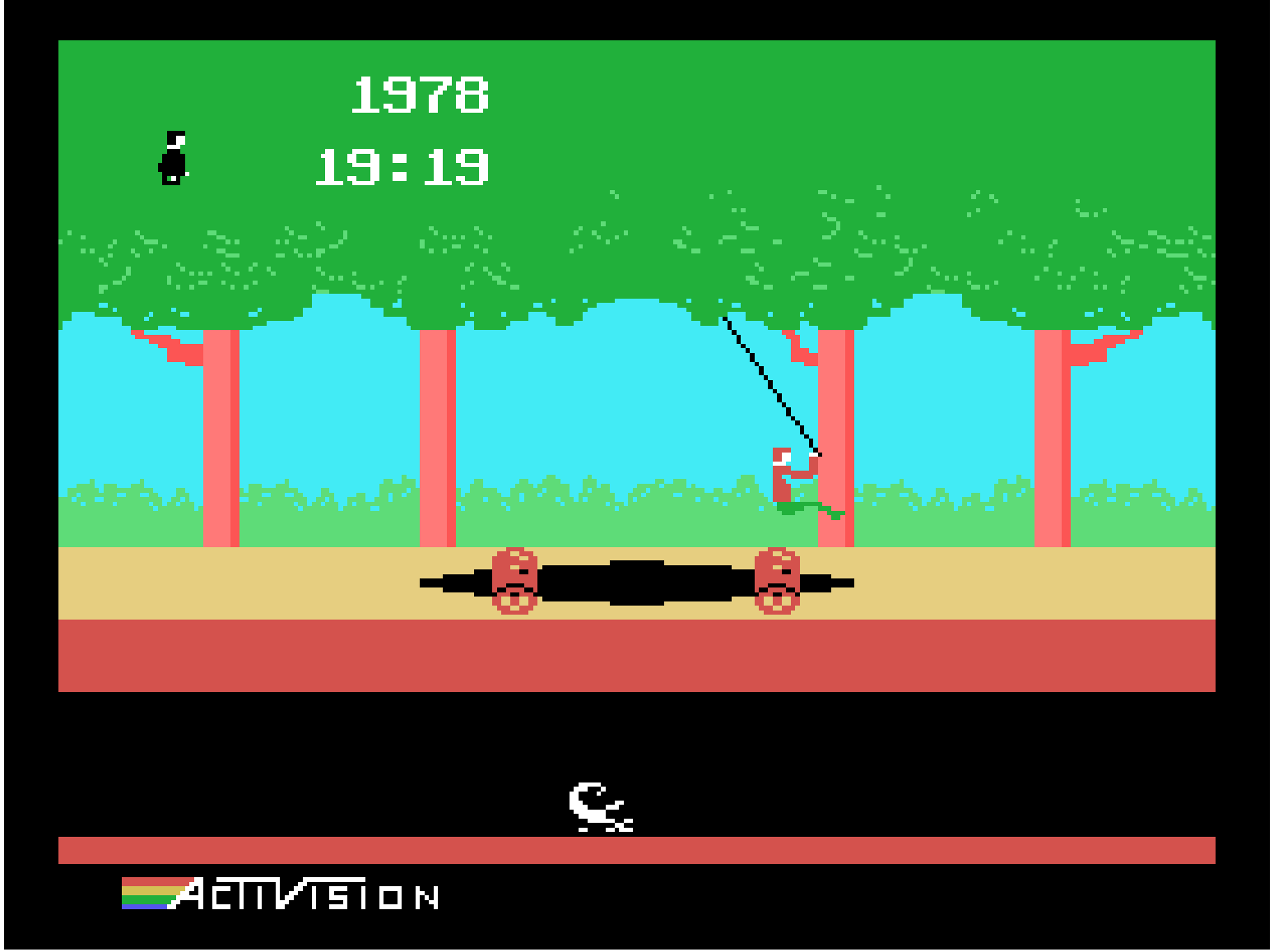

5
10
50
100
(5, 5)
(10, 10)
(50, 50)
(100, 100)

SKLEARN Regression

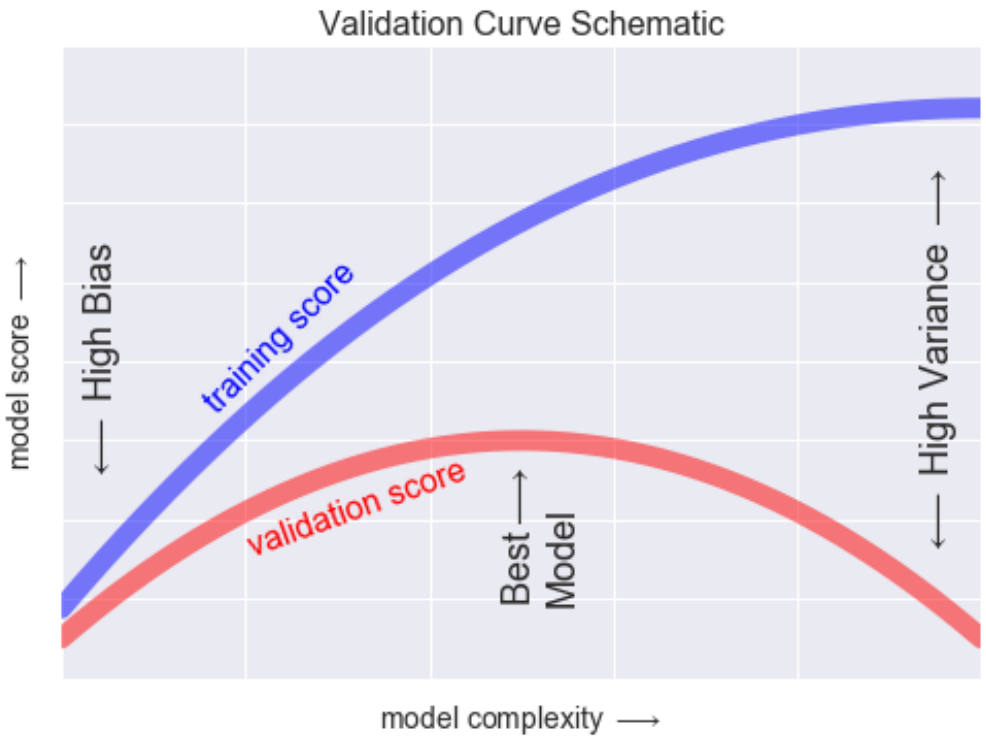
Name	Parameters	Notes
LinearRegression	None	Ordinary least squares, linear algebra solution
Ridge, RidgeCV	L2 (alpha)	Least Squares + L2,
Lasso, LassoCV	L1	Least Squares + L1, better for eliminating features
Elastic	L1, L2	Ridge + Lasso
SGDRegressor	L1 and/or L2	General Purpose
RANSAC		Robust to outliers
SVR	Kernal (non-linear!), L2	Non-linear kernals available
MLPRegressor	L2, hiddens	Most complex model
GradientBoostingRegressor	n_estimators	Very powerful and fast!

L1 gives sparse solutions to eliminate features
L2 keeps them all, adds stability, can be solved analytically or with linalg.

PITFALLS



http://nbviewer.jupyter.org/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.03-Hyperparameters-and-Model-Validation.ipynb



The diagram shown here is often called a *validation curve*, and we see the following essential features:

•The training score is everywhere higher than the validation score. This is generally the case: the model will be a better fit to data it has seen than to data it has not seen.

•For very low model complexity (a high-bias model), the training data is under-fit, which means that the model is a poor predictor both for the training data and for any previously unseen data.

•For very high model complexity (a high-variance model), the training data is over-fit, which means that the model predicts the training data very well, but fails for any previously unseen data.

•For some intermediate value, the validation curve has a maximum. This level of complexity indicates a suitable trade-off between bias and variance.

Pitfalls!

Two common traps to avoid:

- Bias/Variance:
  - Underfitting – your problem isn't linear.
  - Overfitting – common, big problem
- Correlated Features

For more see:
https://scikit-learn.org/stable/auto_examples/inspection/plot_linear_model_coefficient_interpretation.html#sphx-glr-auto-examples-inspection-plot-linear-model-coefficient-interpretation-py

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_regression.html#sklearn.datasets.make_regression

sklearn.datasets.make_regression

sklearn.datasets.make_regression(*n_samples=100, n_features=100, *, n_informative=10, n_targets=1, bias=0.0, effective_rank=None, tail_strength=0.5, noise=0.0, shuffle=True, coef=False, random_state=None*) [source]

Generate a random regression problem.

The input set can either be well conditioned (by default) or have a low rank-fat tail singular profile. See `make_low_rank_matrix` for more details.

The output is generated by applying a (potentially biased) random linear regression model with `n_informative` nonzero regressors to the previously generated input and some gaussian centered noise with some adjustable scale.

Read more in the User Guide.

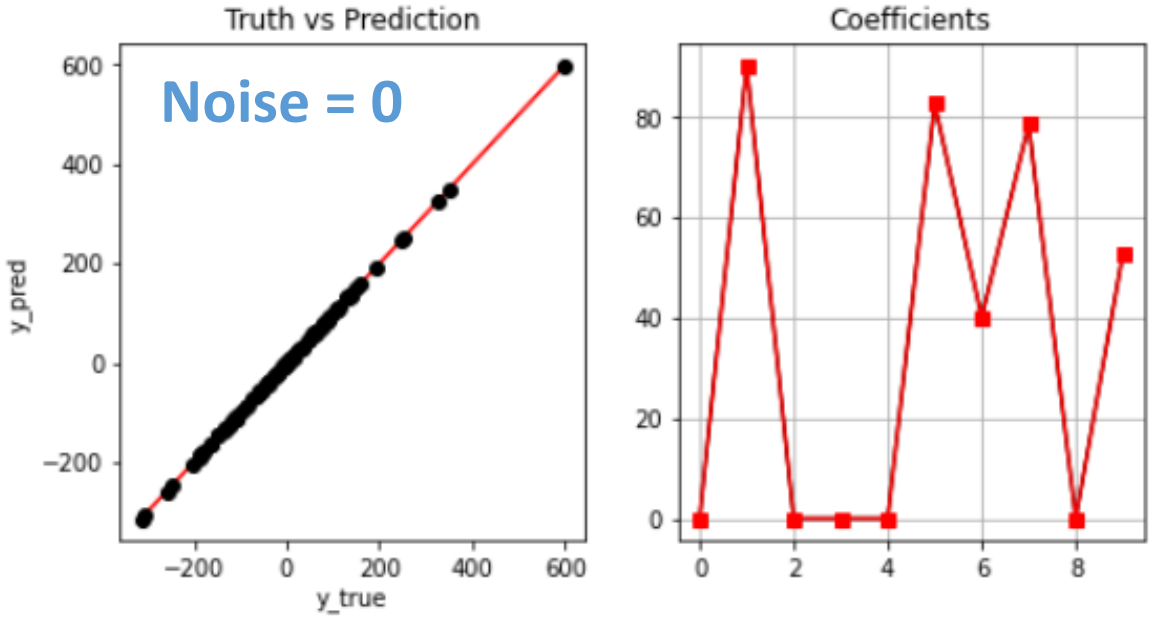
Experiment 0:

100 samples

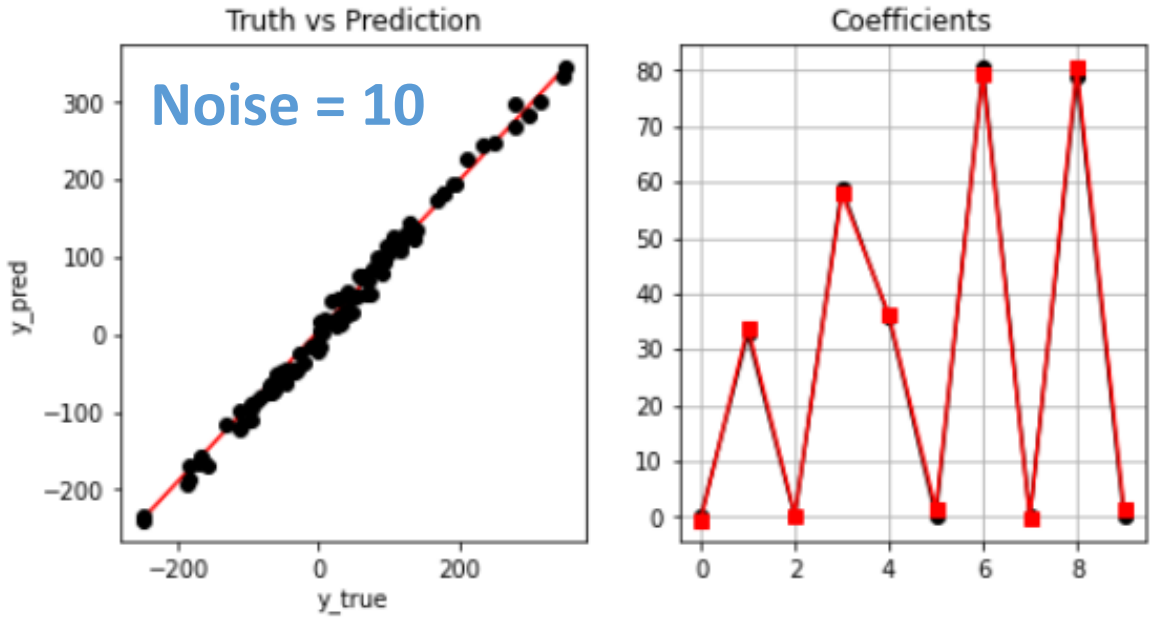
10 features: 5 informative and 5 random

Simple Least Squares

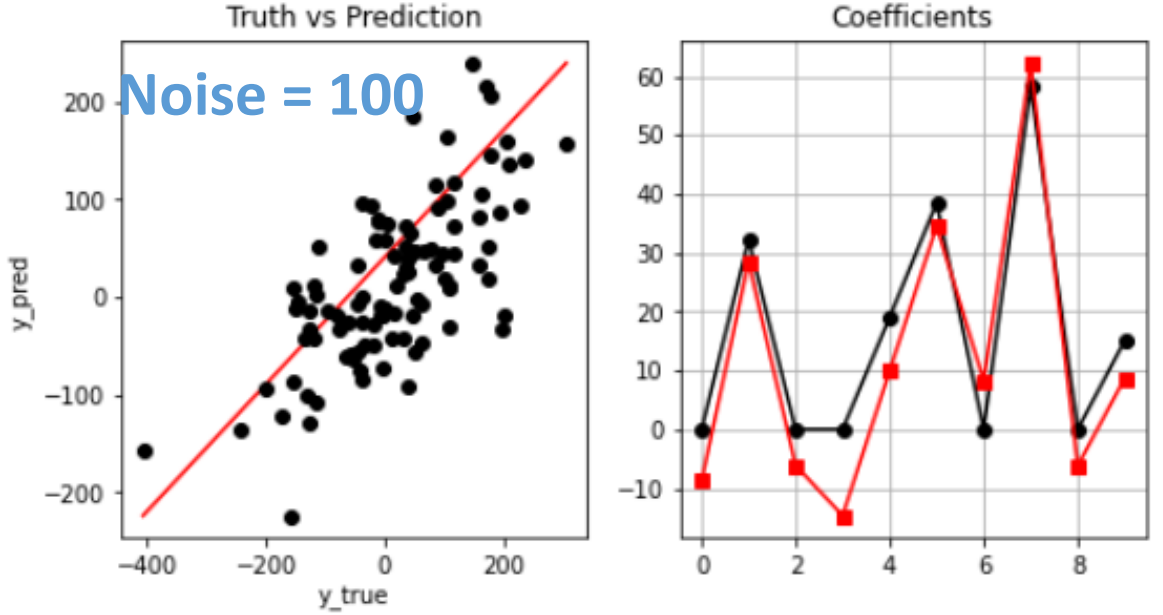
Score 1.0



Score 0.9930626630932599



Score 0.4790240081124104



```python
X,y,coef = make_regression(n_samples=100, n_features=10, n_informative=5, coef=True, noise=100)
print(coef)
```

```
[85.28  0.   14.25  0.    9.67  0.    0.   53.9   0.   98.5 ]
```

```python
reg = LassoCV()
reg.fit(X,y)
y_pred = reg.predict(X)

#plt.figure(figsize=(8,4))
fig, axs = plt.subplots(ncols=2, figsize=(8, 4))
plt.subplot(1,2,1)
PredictionErrorDisplay.from_predictions(y, y_pred, kind="actual_vs_predicted", ax=axs[0])
plt.grid()

plt.subplot(1,2,2)
plt.plot(reg.coef_, '--bs', label='LassoCV')
plt.plot(coef, '-ro', label='Actual')
plt.legend()
plt.title('Coefficients')
plt.grid()

plt.show()
```