Linear Regression and Regularization

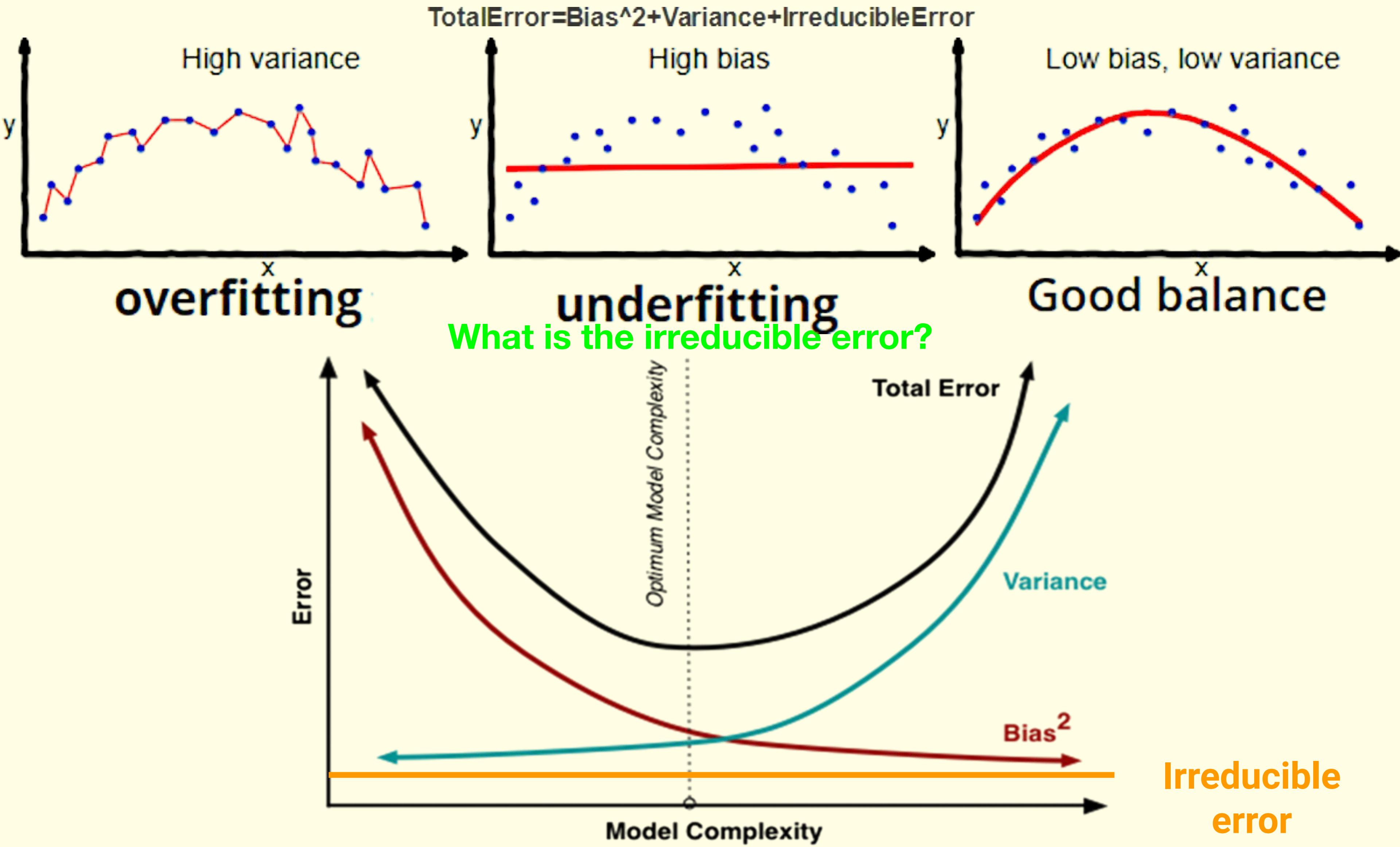
by Dane Brown

“Life is ten percent what you experience and ninety percent how you respond to it.”

Bias-Variance Tradeoff

- **Bias** is the difference between the average prediction of a model and the correct value which it is trying to predict.
- **Variance** is the sensitivity to training data determining how well a model generalizes on unseen data
- Low bias low variance – is this something we want?
- High **bias** means _______fitting
- High **variance** means _______fitting

Reminder about Optimal Fit



Irreducible error

What is the irreducible error?

- Part of a model’s total error that can’t be explained by bias or variance.
- Represents inherent randomness or noise – unrelated to the model.
- Factors that cause this:
  - measurement **errors**
  - natural variability is uncaptured due to measurement **limitations**
  - **unknown** confounding **variables** – those not included in a model

- Irreducible error can be estimated by comparing a model's performance to the Bayes error rate – minimum error any model can achieve with infinite data

Homework: Read the theory slides

Read the following slides and brush up on James' theory

Scoring a Regression Model

- ***`mean_squared_error`*: squared error between **predicted** and **true value** for every data point in the training set, averaged across all data points.

- ***`explained_variance_score`*: the degree a model can explain the variation or dispersion of test data.

- ***`r2_score`*: R^2 score is the quotient of explained variance and total variance for unbiased variance estimation
  - Coefficient of determination
  - Goodness of fit

- They are all poor in the presence of outliers.

Scoring a Regression Model

- ***Mean squared error:***
- MSE = 1/n * ∑(y_pred - y_true)^2

- where:
- n is the number of data points in the training set
- y_pred(ŷ) is the predicted value of the dependent variable for a given data point
- y_true (y) is the true value of the dependent variable for the same data point
- The mean squared error represents the average squared error between the predicted and true values for every data point in the training set.

Scoring a Regression Model

- ***Explained variance score:***
- EVS = 1 - var(y_true - y_pred) / var(y_true)

- where:
- var is the variance of the argument
- y_pred(ŷ) is the predicted value of the dependent variable for a given data point
- y_true (y) is the true value of the dependent variable for the same data point
- The explained variance score represents the degree to which a model can explain the variation or dispersion of the test data. It ranges from 0 to 1, with higher values indicating better performance.

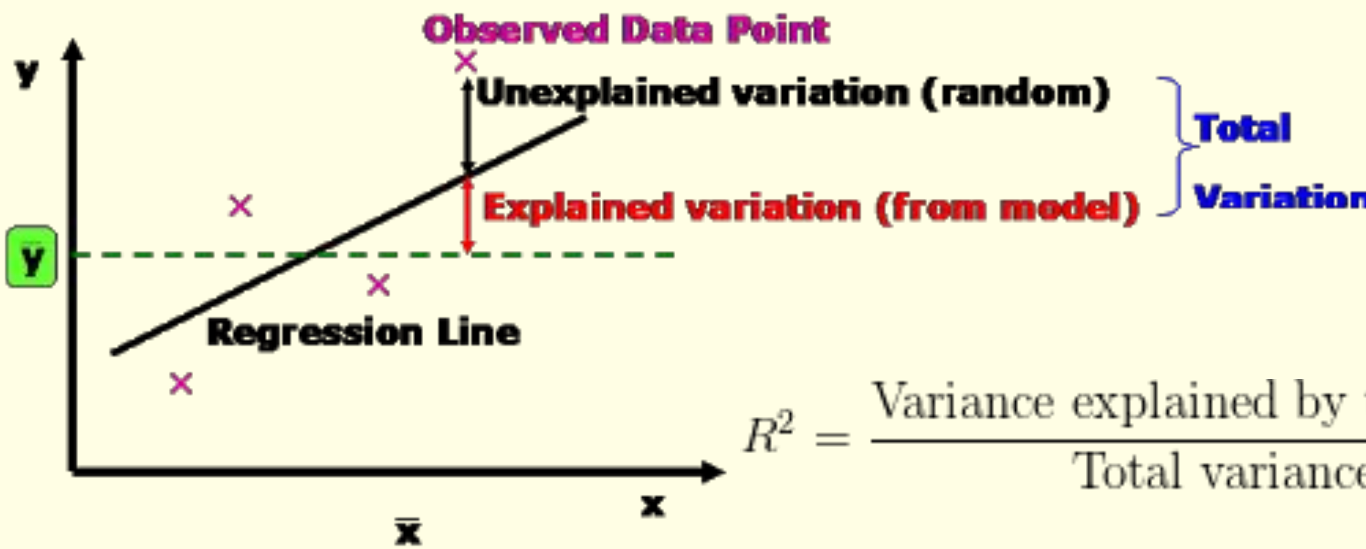
Scoring a Regression Model

- ***R2 score:***
- R2 = 1 - (SS_res / SS_tot)

- where:
- SS_res is the residual sum of squares (unexplained variation)
- SS_tot is the total sum of squares (total variation)

Scoring a Regression Model: R^2 Score

- Distance between the **data point** and the **mean *y value*** is the *total variance*
- Distance between the **mean *y value*** and the **regression line** is the *magnitude of the bias/offset*
- Distance between the **data point** and the **regression line** is the *unexplained variance*

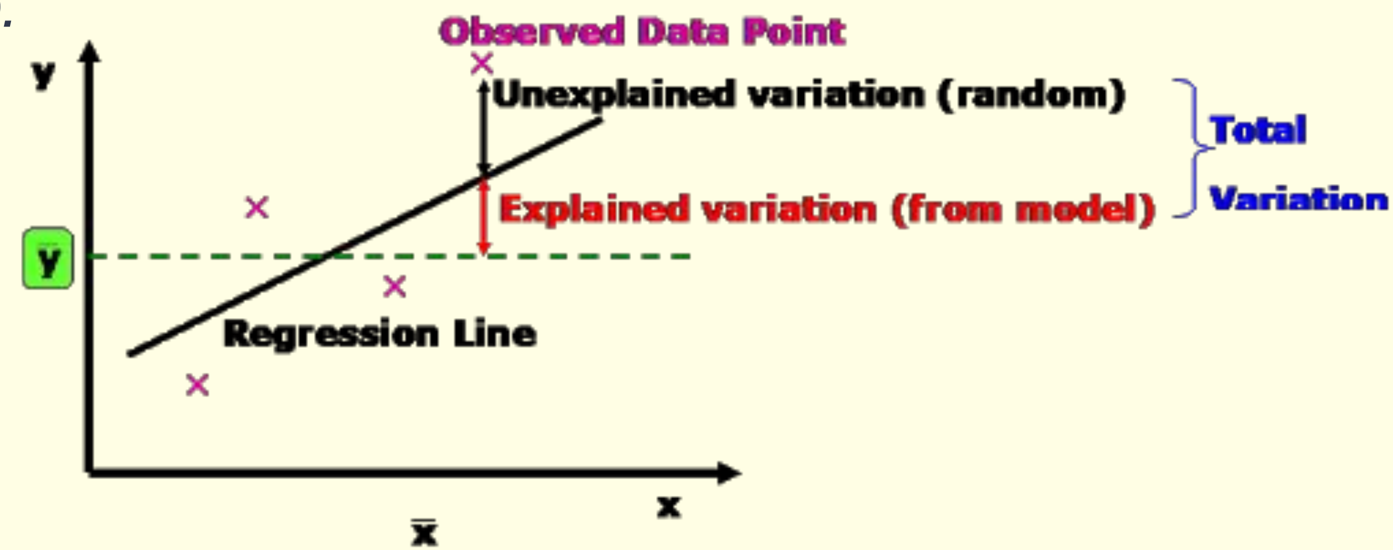


$$R^2 = \frac{\text{Variance explained by the model}}{\text{Total variance}}$$

Scoring a Regression Model: R^2 Score

- explained variation (**SSreg**) = $(\hat{y} - \bar{y})^2$
- unexplained variation (**SSres**) = $(y - \hat{y})^2$
- total variation (**SStot**) = explained variation + unexplained variation
- total variation (**SStot**) = $(y - \bar{y})^2$

Therefore, the closer the R-squared value is to 1, the better the model fits the data and the more of the total variation in y is explained by the independent variable(s).



cvML Methodology

- **Initialization:** Call the `cv` or `scikit` model by name to create an empty instance of the model
- **Set parameters:** default
- **Train the model:** `train` or `fit` is used to fit the model to some data
- **Predict new labels:** use `predict`, to guess the labels of new (**unseen**) data
- **Score the model:** works for both `cv` or `scikit`

Linear Regression in a Nutshell Example

- Target type: **Continuously** predict new outcomes
- Describe a **target** variable with a linear combination of features

- Dataset: Boston house pricing
  - simply predict housing prices
  - no classifying of labels (into classes)
  - $\hat{y} = w_1 f_1 + w_2 f_2$
  - if f_1 and f_2 are today's price for two houses
  - train w_1 and w_2 to learn future price (**target)**
- OpenCV **does not** offer any good implementation of linear regression...

Linear Regression

The structure of the Scikit built-in datasets:

- **DESCR**: Get a description of the data
- **data**: The actual data, num_samples x num_features
- **feature_names**: The names of the variables
- **target**: The class labels, num_samples x 1 **(why?)**
- **target_names**: The names of the class labels, e.g. flower class such as Versicolour

Linear Regression

Check it out -> CV_ML

- The goal is to predict the value of homes in several Boston neighbourhoods in the 1970s, using information as features such as:
  - crime rate
  - property tax rate
  - distance to employment centers
  - highway accessibility
  - etc.

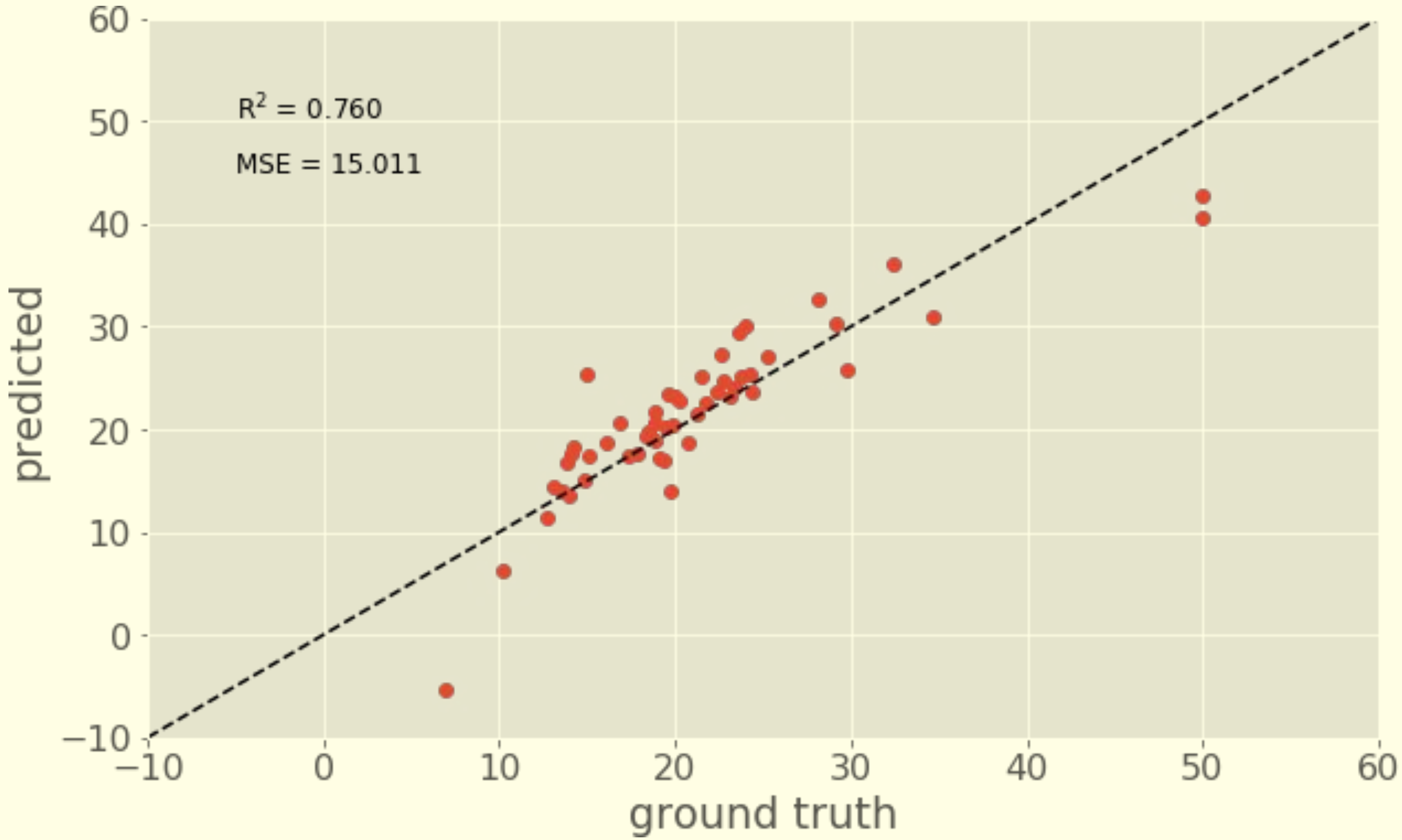
Linear Regression Results

The model is noticeably off at certain points...

Linear Regression Results

- The model tends to be off the most when:
  - i.e. really high or really low housing prices (peaks)
  - peak values of data points 12, 18, and 42.

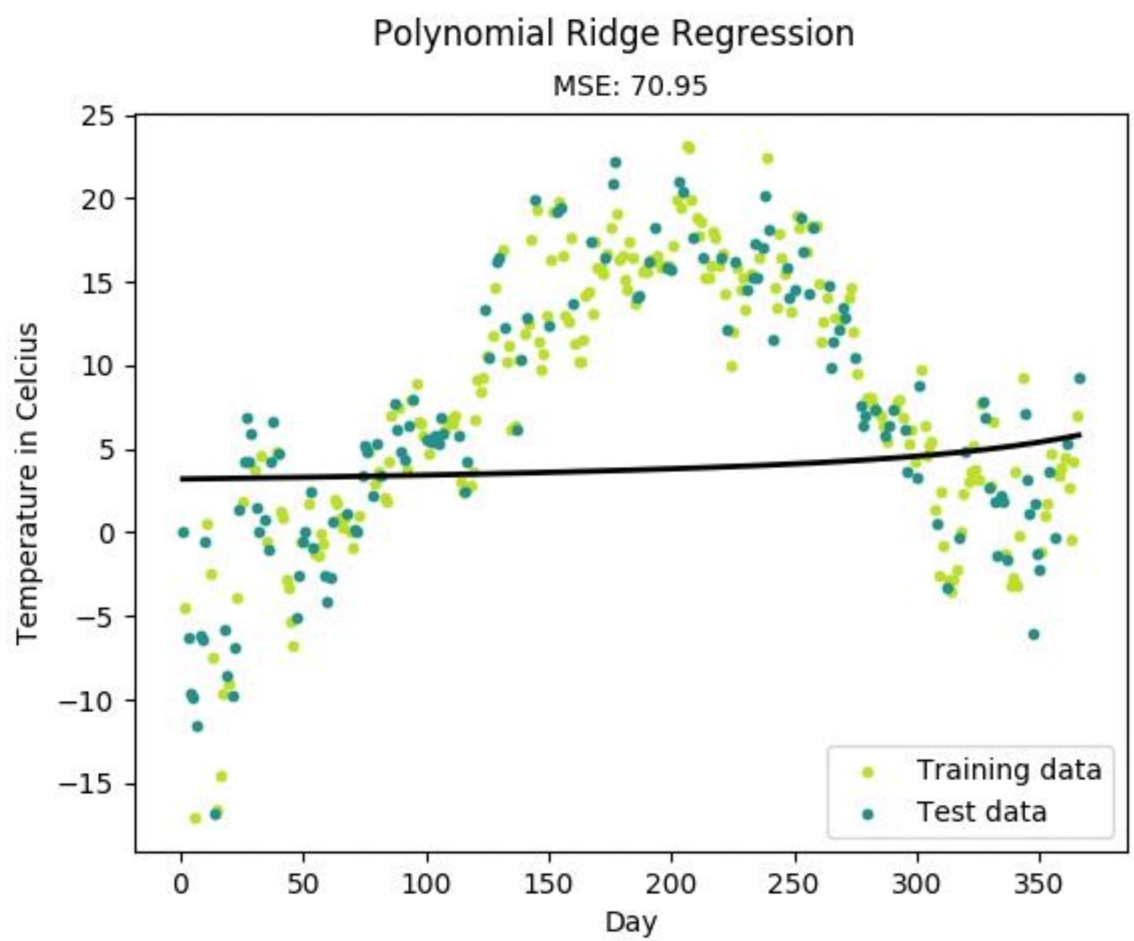
Linear Regression Results



Linear Regression Results

- A perfect model has all data points on the dashed diagonal, since **y_pred** would equal **y_true**
- Deviations from the diagonal indicate model errors: the model was not able to explain some variance in the data .
- R-squared shows that the model explained 76% of the scatter in the data, with an MSE of 15.011.
- ML project: These are the kind of numbers to use in comparisons and results chapter of your thesis!

A “Better” Fit



Polynomial Ridge Regression
MSE: 70.95
Temperature in Celcius
Day
25
20
15
10
5
0
−5
−10
−15
0
50
100
150
200
250
300
350
Training data
Test data

Underfitting

- Easy to spot: Our model does not fit the data well enough and typically has low accuracy on *all* data
  - Little data to build an accurate model on a simple algorithm
  - When we build a linear model with a non-linear data
  - This is covered more explicitly with SVMs
  - Can be fixed without adding more data by selecting or engineering more appropriate features

Overfitting

- An algorithm might work really well on the training *or validation set*, but poorly on test set
- This means the model does not generalize well for unseen data
- This is especially common in decision trees
- This can also happen by overtraining including classification and regression problems

Overfitting: Fixing the data

- Change sampling technique to better estimate model parameters, e.g. k-fold cross validation in a grid search
- Hold back a larger validation dataset

We’ll check out more in later lectures

Overfitting: Reduce it for regression

- Use **Regularization**
- **L1 norm** of the Manhattan distance:
  - This adds a term to the scoring function that is proportional to the sum of all absolute weight values
  - This is used by **Lasso** regression
- **L2 norm** of the Euclidean distance:
  - This adds a term to the scoring function that is proportional to the sum of all squared weight values.
  - Removes strong outliers in the weight vector much more than the L1 norm
  - This is used by **Ridge** regression

Lasso vs. Ridge regularization

Mathematical difference: absolute vs. squared weight

```
L1 = sum (|x(y_true) - x(y_pred)|)

L2 = sum ((x(y_true) - x(y_pred)**2)

MAE = (sum (|x(y_true) - x(y_pred)|))

MSE = (sum ((x(y_true) - x(y_pred)**2))
```

Regularization vs. not

- Original Linear regression scikit function:
  - linreg = linear_model.LinearRegression()

Replace with one of the following:

- Lasso regression
  - linreg = linear_model.Lasso()

- Ridge regression
  - linreg = linear_model.Ridge()

Lasso vs. Ridge

Each ***tend*** to do well under conditions:

- **Lasso**
  - i.e. when few **predictors** influence the response
- **Ridge**
  - i.e. when most **predictors** impact the response

- The boston dataset uses all **predictors** to get prices, but in practice:
- Usually **cross-validation** to select the more suited **features** for a specific case is better
- Terminology check, page over

Lasso vs. Ridge

- **variables / regressors / covariates / predictors**, etc
- Statistics nomenclature that depend on context
- We machine learning blokes call this input or output **features** depending on the context

- **predictors** actually refer to independent variables that affect the output/**target**/response
  - predicted price variables are **not** predictors
- **covariates** are observed continuous predictors
  - depends on context but usually **not** categorical