Linear Models and Evaluation Metrics

IMLP 2.3
IMLP 5
APM Ch
APM Ch 1 12

Table of Contents

Linear Models

- Linear Models have been very widely used and have roots going back hundreds of years
- They predict using a linear equation

Linear Models for Regression

In linear regression we assume that there is a model given by the following linear equation that can give us predicted values $\hat{y}$

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \ldots\ldots + \beta_p * x_p$$

where x denotes the features of a single data point p is the number of predictors and $[\beta_0\ \beta_1\ \beta_2\ \ldots\beta_p]$ are parameters of the model that are to be learned

For a dataset with a single feature this is

$$\hat{y} = \beta_0 + \beta_1 * x_1$$

which is the equation of a straight line with β_1 as `slope` and β_0 as `y-intercept`.

Finding β : Bias Variance Trade-off

If we assume that the data points are statistically independent and that the residuals (y- $\hat{y}$) have a theoretical mean of zero and a constant variance of σ^2 (after about 4 to 5 pages of derivation…)

$$E[\text{MSE}] = \sigma^2 + (\text{Model Bias})^2 + \text{Model Variance},$$

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

σ^2 is called irreducible noise (cannot be changed)
Model Bias – is how far model is from the true relationship between features and target
Model Variance – variance of the model

Bias Variance Trade Off

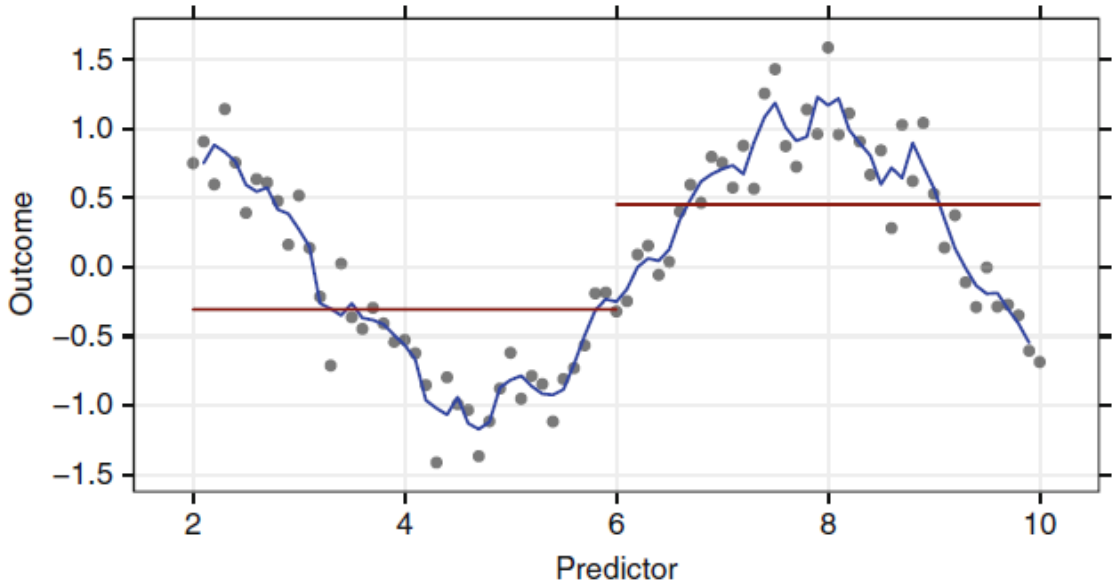


Fig. 5.2: Two model fits to a *sin* wave. The *red line* predicts the data using simple averages of the first and second half of the data. The *blue line* is a three-point moving average

Red Line Fit – splits the data in half – low variance since it would not substantially change if another set of data points were generated the same way- LOW variance and However it is ineffective at modeling the data since due to its simplicity and for this reason it has high bias HIGH bias – LOW Variance HIGH Bias Under fitting.

Blue Line Fit – average of 3 points – It is flexible enough to model the *sin* wave (i.e. low bias) but small perturbations in the data will significantly change the model fit. Because of this it has high variance. HIGH Variance LOW Bias - Overfitting

Measuring Performance of Linear Regression- RMSE

Given that Mean Square Erro

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

Root Mean Square Error is a performance metric for regression

$$\mathrm{RMSE} = \sqrt{MSE}$$

- RMSE can range from 0 to ∞.
- Since the errors are squared before they are averaged the RMSE gives a relatively high weight to large errors.
- This means the RMSE is most useful when large errors are particularly undesirable.

Correlation of determination- R^2

R^2 is most common measure (also wrongly called accuracy in scikit-learn) and it is the square of *correlation* between predicted value and target value. Note that R^2 is *not accuracy* it is *correlation*. If R^2 is .75 it means that model can *explain* 75% of its prediction!

In scikit-learn you can get R^2 by

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

Linear Regression Models

There are several linear models for regression

- Ordinary Least Squares
- Partial Least Squares (SKIP)
- Penalized Least Squares (Lasso and Ridge)

Each of the model will minimize either sum of squares between predicted value and target value or a function of the sum of the squares between predicted and target value.

Ordinary Least Squares

- Linear regression can be solved by multiple ways the most common technique is ordinary least squares is the simplest and most classic linear method for regression
- Solve for coefficients of the equation $\hat{y} = \beta_0 + \beta_1 * x_1 + \ldots + \beta_p * x_p$ to minimize SSE (sum of square errors)

$$E[\text{MSE}] = \sigma^2 + (\text{Model Bias})^2 + \text{Model Variance},$$

- Ordinary Least Squares minimizes the *bias* term of the equation above.
- It has NO model parameters which makes the model simple

Linear Regression-Ordinary Least Squares

The objective of ordinary least squares linear regression is to find the plane that minimizes the sum-of-squared errors (SSE) between the observed and predicted response:

$$\mathrm{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where y_i is the outcome and $\hat{y}_i$ s the model prediction of y_i given by the following.

$\hat{y}_i = \beta_0 + \beta_1 * x_{i1} + \ldots . + \beta_p * x_{iP}$

This can be written as $X . \beta = \hat{y}$ where $\beta = [\beta_0 \ \beta_1 \ \beta_2 \ \ldots \beta_p]^T$

and $X = \begin{bmatrix} 1 & x_{11} & \ldots & x_{1P} \\ 1 & x_{21} & \ldots & x_{2P} \\ .. & \ldots & .. & .. \\ 1 & x_{n1} & \ldots & x_{nP} \end{bmatrix}$

Mathematically the optimal plane β can be shown to be

$$(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y,$$

where **X** is the matrix of predictors P along with a first column of 1s as shown above and *y* is the target vector.

Ordinary Least Squares – Inverse Issue

A unique inverse matrix $(X^T X)^{(-1)}$ exists when

(1) no predictor can be determined from a combination of one or more of the other predictors and (highly correlated features). Each column should be independent of other columns

(2) the number of samples is greater than the number of predictors

If the model requirements above are not met then

1) by either replacing $(\mathbf{X}^T\mathbf{X})^{-1}$ with a conditional inverse (loses interpretability of predictors) (scikit-learn uses this technique called pseudo-inverse)
2) by removing predictors that are collinear

Correlation coefficient between two vectors of values x and y

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

$\bar{x}$ is the mean of the values in vector x and $\bar{y}$ Is the mean of the values in vector y

You will get a number between -1 to 1. Positive indicates highly correlated negative indicates negatively correlated and close to 0 indicates no correlation .

- You can use Numpy's `corrcoef` function to compute correlation between two features
- This is called pearson's correlation

Correlation between two variables

Use “heatmap” to find correlation between each pair of features. In pandas you can call corr() function.

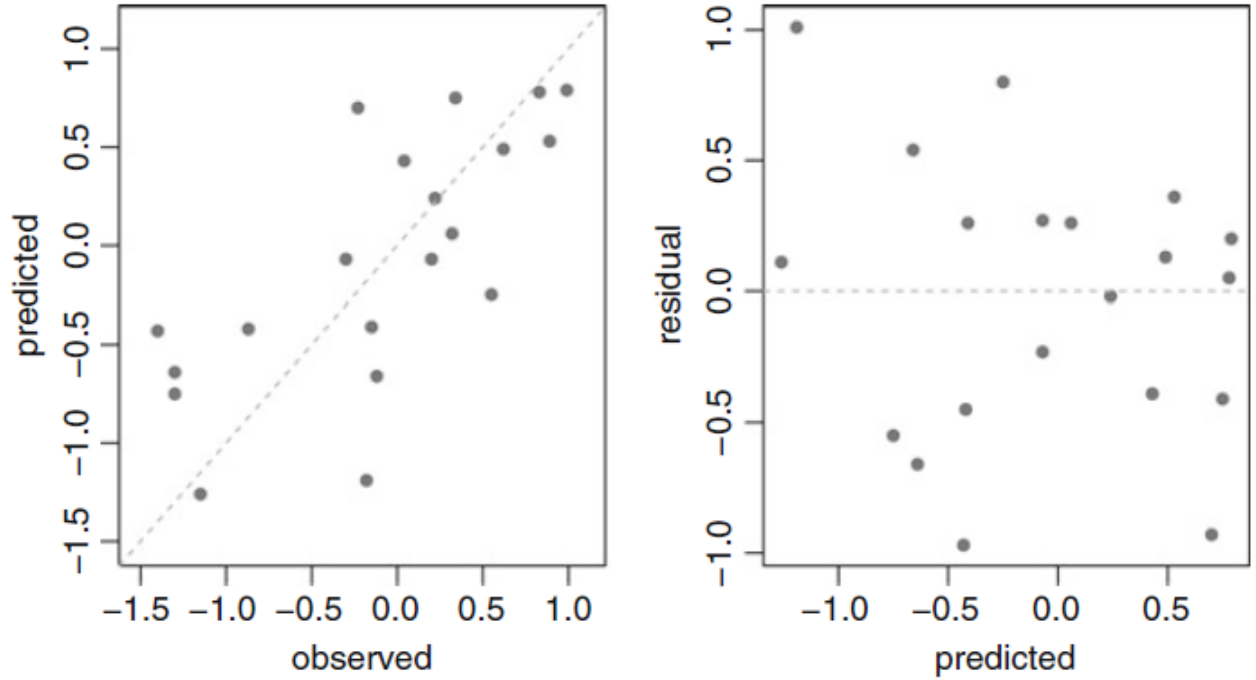
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
CRIM	1.000000	-0.199458	0.404471	-0.055295	0.417521	-0.219940	0.350784	-0.377904	0.622029	0.579564	0.288250	-0.377365	0.452220	-0.385832
ZN	-0.199458	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.360445
INDUS	0.404471	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.483725
CHAS	-0.055295	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.175260
NOX	0.417521	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.427321
RM	-0.219940	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	0.695360
AGE	0.350784	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	-0.376955
DIS	-0.377904	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	0.249929
RAD	0.622029	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	-0.381626
TAX	0.579564	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	-0.468536
TRATIO	0.288250	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044	-0.507787
B	-0.377365	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087	0.333461
LSTAT	0.452220	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000	-0.737663
PRICE	-0.385832	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737663	1.000000

output of bostan.corr()

Ordinary Least Squares- What if relation is non-linear?

Curvature in the predicted-versus-residual plot is a primary indicator that the underlying relationship is not linear.

- If it is known to be quadratic or cubic etc those can be added to the features



Note that trying to fit a linear model to the non-linear data you see residual is far away from 0.

Ordinary Least Squares – Outlier data

- A third notable problem with multiple linear regression is that it is prone to chasing observations that are away from the overall trend of the majority of the data.
- Recall that linear regression seeks to find the parameter estimates that minimize SSE; hence observations that are far from the trend of the majority of the data will have exponentially large residuals.
- In order to minimize SSE linear regression will adjust the parameter estimates to better accommodate these unusual observations.
- Observations that cause significant changes in the parameter estimates are called *influential* and the field of robust regression has been developed to address these kinds of problems.

Effect of residual to the objective function

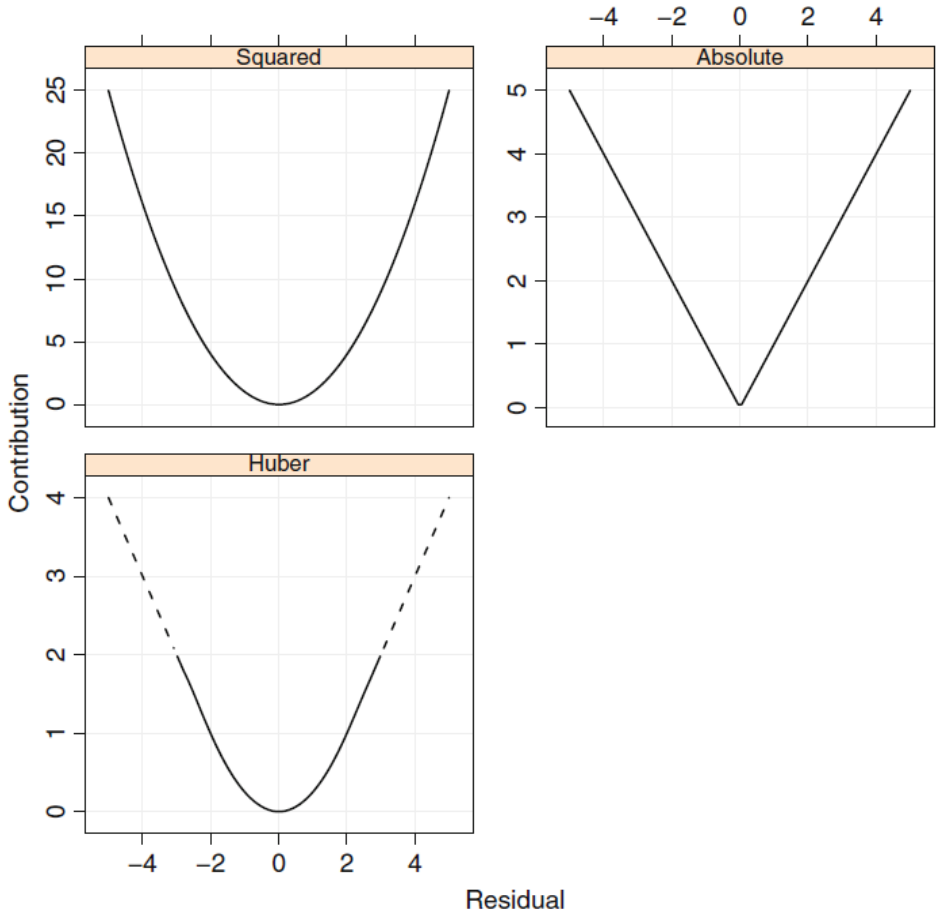


Fig. 6.6: The relationship between a model residual and its contribution to the objective function for several techniques. For the Huber approach, a threshold of 2 was used

Also the Huber function uses the squared residuals when they are “small” and the simple different between the observed and predicted values when the residuals are above a threshold

Linear Regression Model for Wave Dataset

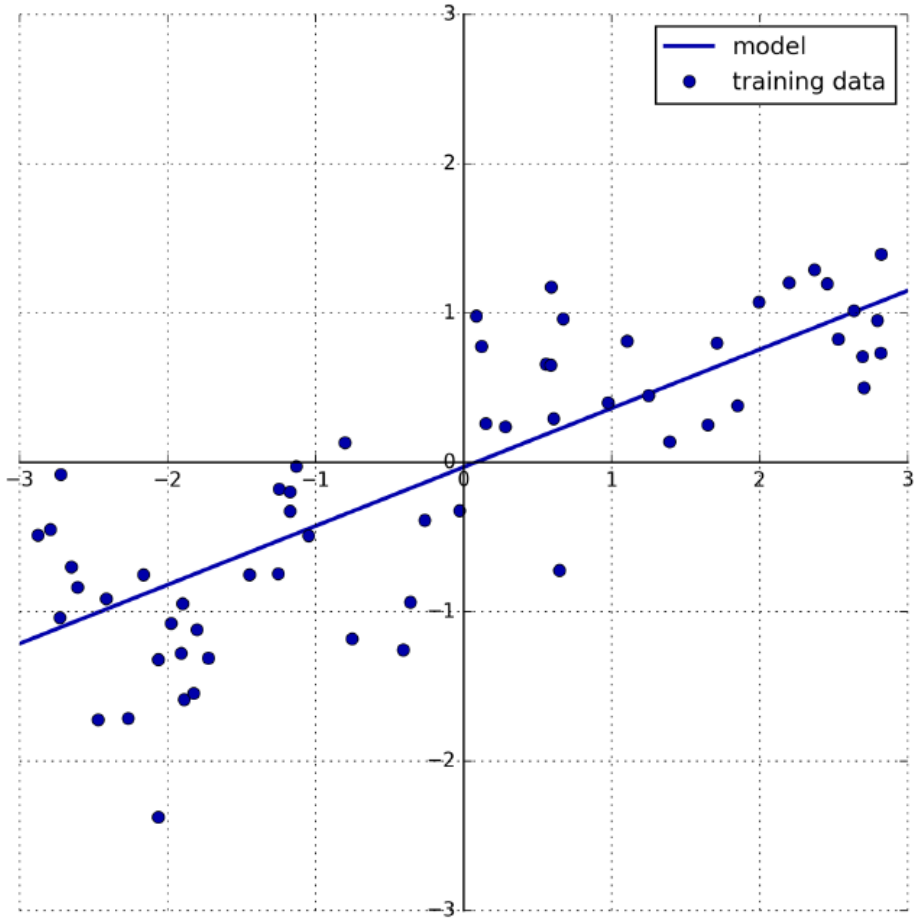


Figure 2-11. Predictions of a linear model on the wave dataset

Computation

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression().fit(X_train y_train)
```

The "slope" parameters (*w*) also called weights or *coefficients* are stored in the coef_ attribute while the offset or *intercept* (*b*) is stored in the intercept_ attribute:

```
print("lr.coef_:" lr.coef_)

print("lr.intercept_:" lr.intercept_)

print("Training set score: {:.2f}".format(lr.score(X_train y_train)))

print("Test set score: {:.2f}".format(lr.score(X_test y_test)))
```

```
lr.coef_: [0.394]

lr.intercept_: -0.031804343026759746

Training set score: 0.95

Test set score: 0.61
```

Penalized Linear Regression-Ridge regression

- Ridge regression is also a linear model for regression so the formula it uses to make predictions is the same one used for ordinary least squares
- In ridge regression though the coefficients are chosen not only so that they predict well on the training data but also to fit an additional constraint i.e least value.
- Intuitively this means each feature should have as little effect on the outcome as possible while still predicting well
- Regularization means explicitly restricting a model to avoid overfitting
- The particular kind used by ridge regression is known as L2 regularization
- The Ridge model makes a trade-off between the simplicity of the model and its performance on the training set

Penalized Linear Regression-Ridge regression

$$\mathrm{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

$$\mathrm{SSE}_{L_2} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} \beta_j^2.$$

$\hat{y} = \beta_0 + \beta_1 * x_1 + \ldots + \beta_p * x_p$

- L_2 signifies the **squaring** the coefficient terms β
- In effect this method *shrinks* the estimates towards 0 as the *λ* penalty becomes large (these techniques are sometimes called “shrinkage methods”)
- Scikit-learn uses Alpha (α) for *λ*

Boston Housing Dataset

We will be using a real-world regression dataset the Boston Housing dataset. The task associated with this dataset is to predict the median value of homes in several Boston neighborhoods in the 1970s using information such as crime rate proximity to the Charles River highway accessibility and so on. The dataset contains 506 data points described by 13 features:

Ridge Regression - Boston Housing Dataset

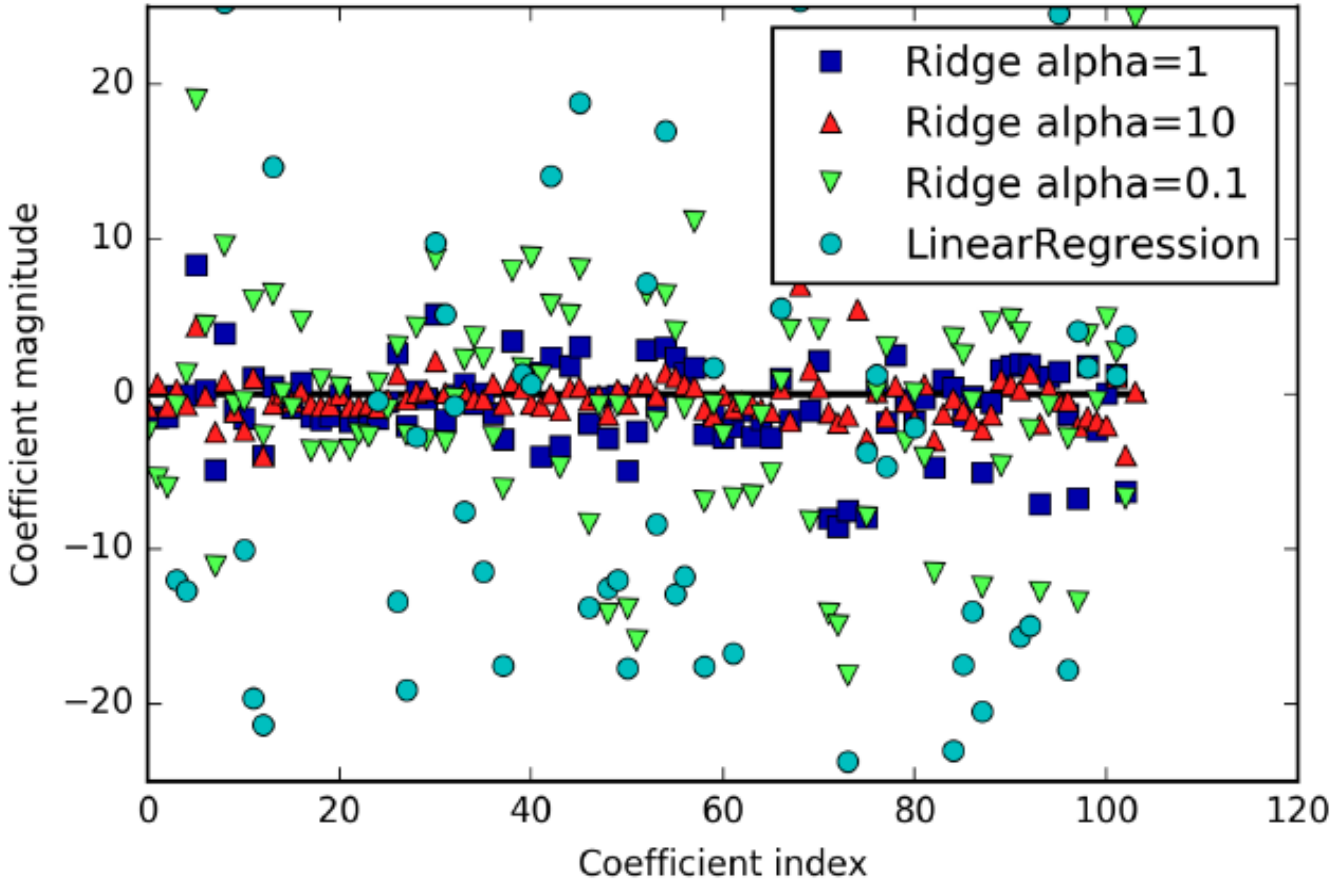


Figure 2-12. Comparing coefficient magnitudes for ridge regression with different values of alpha and linear regression

Ridge Regression Score – Boston Housing Dataset

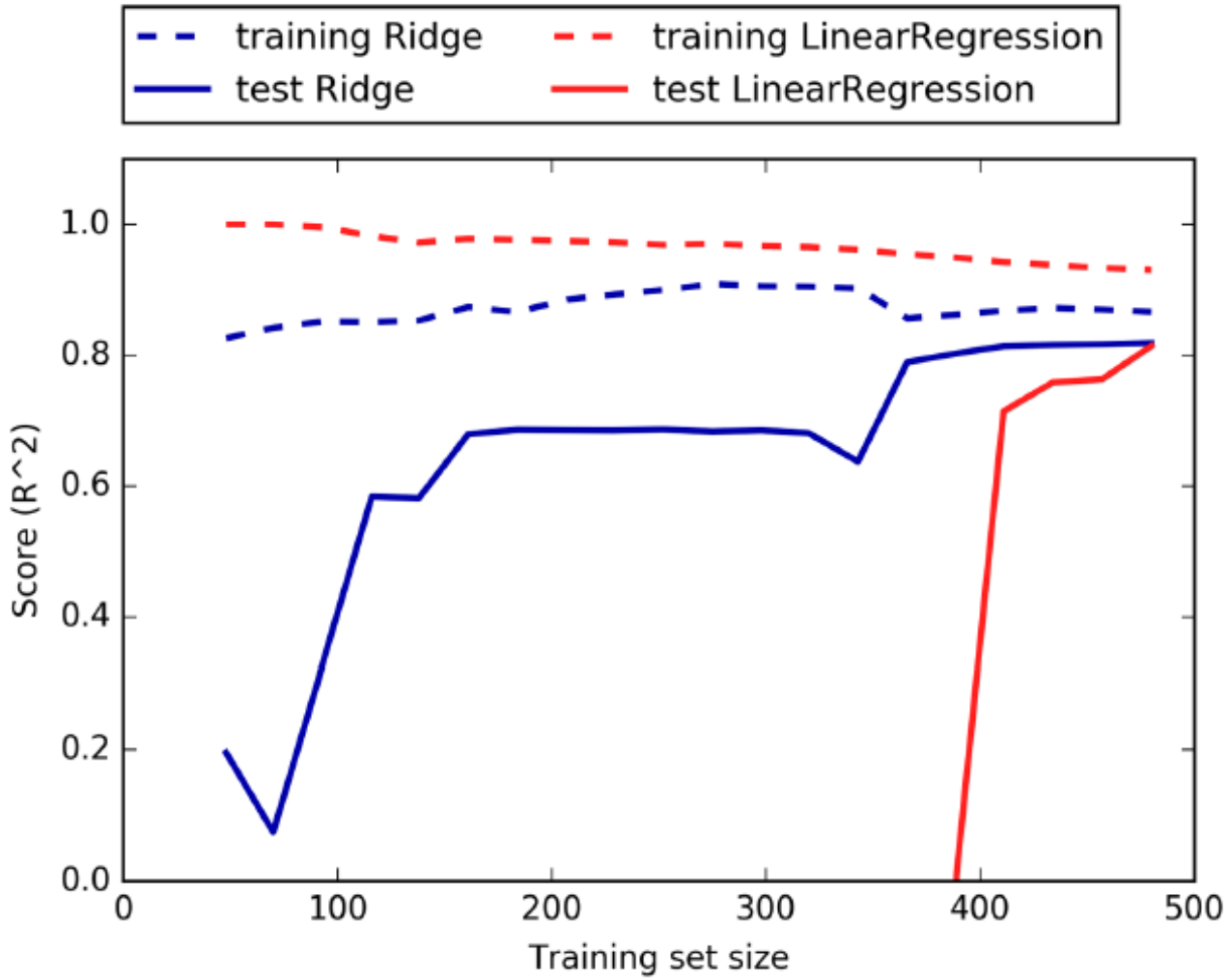


Figure 2-13. Learning curves for ridge regression and linear regression on the Boston Housing dataset

Ridge regression

- How much importance the model places on simplicity versus training set performance can be specified by the user using the alpha parameter
- In the previous example we used the default parameter alpha=1
- The optimum setting of alpha depends on the particular dataset we are using
- Increasing alpha forces coefficients to move more toward zero which decreases training set performance but might help generalization
- NOTE: alpha is 0 means--- there is no shrinkage at all
- The y-axis shows the numeric values of the corresponding values of the coefficients

Ridge regression

- The main takeaway here is that for alpha=10 the coefficients are mostly between around –3 and
- The coefficients for the Ridge model with alpha=1 are somewhat larger
- The dots corresponding to alpha=0.1 have larger magnitude still
- Many of the dots corresponding to linear regression without any regularization are so large they are outside of the chart

Computation

```
from sklearn.linear_model import Ridge
ridge10 = Ridge(alpha=10).fit(X_train y_train)
print("Training set score:
{:.2f}".format(ridge10.score(X_train y_train)))
print("Test set score:
{:.2f}".format(ridge10.score(X_test y_test)))
```

```
Training set score: 0.79
Test set score: 0.64
```

Penalized Linear Regression-Lasso Regression

- An alternative to Ridge for regularizing linear regression is Lasso
- As with ridge regression the lasso also restricts coefficients to be close to zero but in a slightly different way called L1 regularization
- The consequence of L1 regularization is that when using the lasso some coefficients are exactly zero
- This means some features are entirely ignored by the model
- Having some coefficients be exactly zero often makes a model easier to interpret and can reveal the most important features of your model
- No closed form solution exists for minimization

Lasso Regression

$$\mathrm{SSE} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

$$\mathrm{SSE}_{L_1} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} |\beta_j|.$$

A consequence of penalizing the absolute values is that some parameters are actually set to 0 for some value of λ. Thus the lasso yields models that simultaneously use regularization to improve the model and to conduct feature selection

"Ridge regression is known to shrink the coefficients of correlated predictors towards each other allowing them to borrow strength from each other. In the extreme case of *k* identical predictors they each get identical coefficients with 1/*k*th the size that any single one would get if fit alone.[. . .] lasso on the other hand is somewhat indifferent to very correlated predictors and will tend to pick one and ignore the rest."

Lasso Regression on Boston Housing

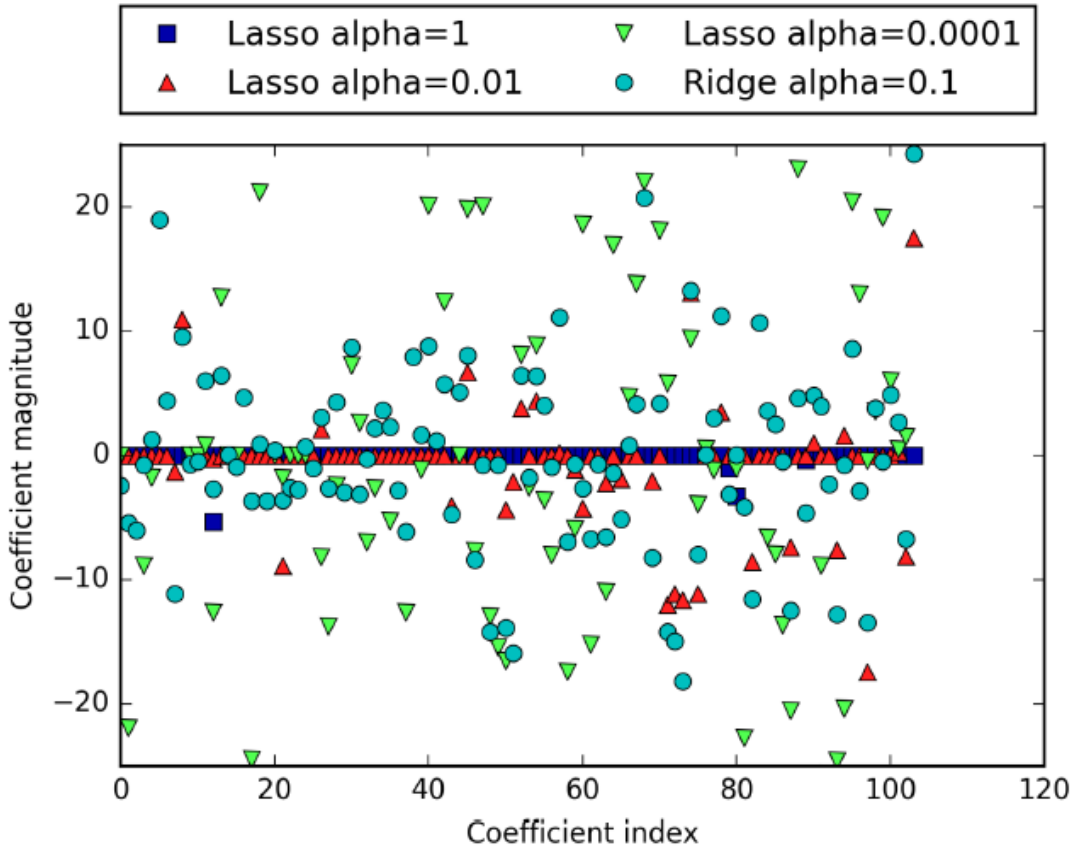


Figure 2-14. Comparing coefficient magnitudes for lasso regression with different values of alpha and ridge regression

Computation

```
from sklearn.linear_model import Lasso
lasso001 = Lasso(alpha=0.01 max_iter=100000).fit(X_train y_train)
print("Training set score: {:.2f}".format(lasso001.score(X_train y_train)))
print("Test set score: {:.2f}".format(lasso001.score(X_test y_test)))
print("Number of features used:" np.sum(lasso001.coef_ != 0))
```

```
Training set score: 0.90
Test set score: 0.77
Number of features used: 33  (out of 104)
```

Class Activity (1)

(1) a) Fit linear model to following dataset

Feature1 (x_1)	y
2	5
1	2

Number of samples > number of features

b) Predict the value of y for [4] and [3]

c) If correct value for [4] is 1 and [3] is 2, find R^2 and RMSE

Class Activity-Solution to (1)

a) The model is $y = \beta_0 + \beta_1 * x_1$

The parameters of the linear model is given by

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y,$$

We need to find β

$$\beta = \begin{bmatrix}\beta_0 \\ \beta_1\end{bmatrix}$$

Where X and y are given as

$$X = \begin{bmatrix}1 & 2 \\ 1 & 1\end{bmatrix} \qquad y = \begin{bmatrix}y_1 \\ y_2\end{bmatrix} = \begin{bmatrix}5 \\ 2\end{bmatrix}$$

$$X^T = \begin{bmatrix}1 & 1 \\ 2 & 1\end{bmatrix} \qquad X^T X = \begin{bmatrix}1 & 1 \\ 2 & 1\end{bmatrix}\begin{bmatrix}1 & 2 \\ 1 & 1\end{bmatrix} = \begin{bmatrix}1*1+1*1 & 1*2+1*1 \\ 2*1+1*1 & 2*2+1*1\end{bmatrix}$$

$$X^T X = \begin{bmatrix}2 & 3 \\ 3 & 5\end{bmatrix}$$

Class Activity-Solution to (1)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$X^T X = \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix}$

$(X^T X)^{(-1)} = \frac{1}{2.5 - 3.3} \begin{bmatrix} 5 & -3 \\ -3 & 2 \end{bmatrix} = \begin{bmatrix} 5 & -3 \\ -3 & 2 \end{bmatrix}$ $\quad X^T = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$ $\quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y, = \begin{bmatrix} 5 & -3 \\ -3 & 2 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 5 \\ 2 \end{bmatrix} = \begin{bmatrix} 5*1-3*2 & 5*1-3*1 \\ -3*1+2*2 & -3*1+2*1 \end{bmatrix}\begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 2 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 5 \\ 2 \end{bmatrix} = \begin{bmatrix} -1*5+2*2 \\ 1*5-1*2 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

The model is $y = -1 + 3 * x_1$

Class Activity-Solution to (1)

b) Predicting the value of y for [4] and [3]

The model is $y = -1 + 3 * x_1$

When x_1 is 4, y = -1 + 3*4 = **11**
When x_1 is 3, y is y = -1 + 3*3 = **8**

c) If correct value for [4] is 10 and [3] is 9, find R^2 and RMSE

$$R^2 = 1 - \frac{\sum (y_{true} - y_{pred})^2}{\sum (y_{true} - y_{true.mean})^2}$$

$y_{true.mean} = (1+2)/2 = 1.5$

$R^2 = 1- [(10-11)^2 + (9-8)^2] / [(1-1.5)^2 + (2-1.5)^2] = 1- (1+1)/(.25 + .25) =1- 2/0.5 = 1- 4 = -3$

RMSE = $\sqrt{MSE}$, MSE = $1/2[(10-11)^2 + (9-8)^2] = 2/2 = 1$

Class Activity

2. Fit linear model to following dataset

Feature1 (x_1)	Feature2 (x_2)	y
2	3	5
1	5	2
4	2	3

Number of samples > number of features

Predict the value of y for [4 5] and [3 4]

Class Activity- Solution (2)

The model is $y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$

The parameters of the linear model is given by

$$\beta = (X^T X)^{-1} X^T y,$$

We need to find β

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

where X is

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 5 \\ 1 & 4 & 2 \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \\ 3 \end{bmatrix}$$

$$X^T = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 4 \\ 3 & 5 & 2 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 4 \\ 3 & 5 & 2 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 5 \\ 1 & 4 & 2 \end{bmatrix} = \begin{bmatrix} 1*1+1*1+1*1 & 1*2+1*1+1*4 & 1*3+1*5+1*2 \\ 2*1+1*1+4*1 & 2*2+1*1+4*4 & 2*3+1*5+4*2 \\ 3*1+5*1+2*1 & 3*2+5*1+2*4 & 3*3+5*5+2*2 \end{bmatrix} = \begin{bmatrix} 3 & 7 & 10 \\ 7 & 21 & 19 \\ 10 & 19 & 38 \end{bmatrix}$$

Class Activity- Solution (2)

$X^T X = \begin{bmatrix} 3 & 7 & 10 \\ 7 & 21 & 19 \\ 10 & 19 & 38 \end{bmatrix}$

Finding inverse $(X^T X)^{(-1)} = \begin{bmatrix} 3 & 7 & 10 \\ 7 & 21 & 19 \\ 10 & 19 & 38 \end{bmatrix}$

Class Activity- Solution (2)

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

Minor of element a is $\begin{bmatrix} e & f \\ h & i \end{bmatrix}$

Minor of element b is $\begin{bmatrix} d & f \\ g & i \end{bmatrix}$

Minor of element c is $\begin{bmatrix} d & e \\ g & h \end{bmatrix}$

Minor of element d is $\begin{bmatrix} b & c \\ h & i \end{bmatrix}$

Minor of element e is $\begin{bmatrix} a & c \\ g & i \end{bmatrix}$

Minor of element f is $\begin{bmatrix} a & b \\ g & h \end{bmatrix}$

Minor of element g is $\begin{bmatrix} b & c \\ e & f \end{bmatrix}$

Minor of element h is $\begin{bmatrix} a & c \\ d & f \end{bmatrix}$

Minor of element i is $\begin{bmatrix} a & b \\ d & e \end{bmatrix}$

Class Activity- Solution (2)

$X^T X = \begin{bmatrix} 3 & 7 & 10 \\ 7 & 21 & 19 \\ 10 & 19 & 38 \end{bmatrix}$ Minor Matrix = $\begin{bmatrix} 437 & 76 & -77 \\ 76 & 14 & -13 \\ -77 & -13 & 14 \end{bmatrix}$

Minor of element 3 is $\det\begin{bmatrix} 21 & 19 \\ 19 & 38 \end{bmatrix}$ = 21*38-19*19 =437 Minor of element 7 is $\det\begin{bmatrix} 7 & 19 \\ 10 & 38 \end{bmatrix}$ = 21*38-19*10=76

Minor of element 10 is $\det\begin{bmatrix} 7 & 21 \\ 10 & 19 \end{bmatrix}$ = 7*19-21*10=-77 Minor of element 7 is $\det\begin{bmatrix} 7 & 10 \\ 19 & 38 \end{bmatrix}$ = 7*38-19*10= 76

Minor of element 21 is $\det\begin{bmatrix} 3 & 10 \\ 10 & 38 \end{bmatrix}$ = 3*38-10*10= 14 Minor of element 19 is $\det\begin{bmatrix} 3 & 7 \\ 10 & 19 \end{bmatrix}$ = 3*19-7*10= -13

Minor of element 10 is $\det\begin{bmatrix} 7 & 10 \\ 21 & 19 \end{bmatrix}$ = 7*19-21*10=-77 Minor of element 19 is $\det\begin{bmatrix} 3 & 10 \\ 7 & 19 \end{bmatrix}$ = 3*19-10*7= -13

Minor of element 38 is $\det\begin{bmatrix} 3 & 7 \\ 7 & 21 \end{bmatrix}$ = 3*21-7*7= 14

Class Activity- Solution (2)

$X^T X = \begin{bmatrix} 3 & 7 & 10 \\ 7 & 21 & 19 \\ 10 & 19 & 38 \end{bmatrix}$

Minor Matrix = $\begin{bmatrix} 437 & 76 & -77 \\ 76 & 14 & -13 \\ -77 & -13 & 14 \end{bmatrix}$

Co-factor sign= $\begin{pmatrix} + & - & + \\ - & + & - \\ + & - & + \end{pmatrix}$,

Co-factor Matrix = $\begin{bmatrix} 437 & -76 & -77 \\ -76 & 14 & 13 \\ -77 & 13 & 14 \end{bmatrix}$

Co-factor Transpose Matrix = $\begin{bmatrix} 437 & -76 & -77 \\ -76 & 14 & 13 \\ -77 & 13 & 14 \end{bmatrix}$

Class Activity- Solution (2)

The **determinant of matrix A** is calculated as

$$\det\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = a \cdot \det\begin{bmatrix} e & f \\ h & i \end{bmatrix} - b \cdot \det\begin{bmatrix} d & f \\ g & i \end{bmatrix} + c \cdot \det\begin{bmatrix} d & e \\ g & h \end{bmatrix}$$

$$\det\begin{bmatrix} 3 & 7 & 10 \\ 7 & 21 & 19 \\ 10 & 19 & 38 \end{bmatrix} = 3 * \det\begin{bmatrix} 21 & 19 \\ 19 & 38 \end{bmatrix} \; -7 * \det\begin{bmatrix} 7 & 19 \\ 10 & 38 \end{bmatrix} \; +10 * \det\begin{bmatrix} 7 & 21 \\ 10 & 19 \end{bmatrix}$$

$= 3 * 437 - 7 * 76 + 10 (-77) = 9$

Class Activity- Solution (2)

$$(X^T X)^{(-1)} = 1/(9) \begin{bmatrix} 437 & -76 & -77 \\ -76 & 14 & 13 \\ -77 & 13 & 14 \end{bmatrix}$$

$$(X^T X)^{(-1)} X^T = 1/(9) \begin{bmatrix} 437 & -76 & -77 \\ -76 & 14 & 13 \\ -77 & 13 & 14 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 4 \\ 3 & 5 & 2 \end{bmatrix}$$

$$1/(9) \begin{bmatrix} 437*1 - 76*2 - 77*3 & 437*1 - 76*1 - 77*5 & 437*1 - 76*4 - 77*2 \\ -76*1 + 14*2 + 13*3 & -76*1 + 14*1 + 13*5 & -76*1 + 14*4 + 13*2 \\ -77*1 + 13*2 + 14*3 & -77*1 + 13*1 + 14*5 & -77*1 + 13*4 + 14*2 \end{bmatrix}$$

$$(X^T X)^{(-1)} X^T = 1/(9) \begin{bmatrix} 54 & -24 & -21 \\ -9 & 3 & 6 \\ -9 & 6 & 3 \end{bmatrix} = \begin{bmatrix} 6 & -8/3 & -7/3 \\ -1 & 1/3 & 2/3 \\ -1 & 2/3 & 1/3 \end{bmatrix}$$

Class Activity- Solution (2)

$(X^T X)^{(-1)} X^T y = \begin{bmatrix} 6 & -8/3 & -7/3 \\ -1 & 1/3 & 2/3 \\ -1 & 2/3 & 1/3 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \\ 3 \end{bmatrix}$

$(X^T X)^{(-1)} X^T y = \begin{bmatrix} 6*5 - 8/3*2 - 7/3*3 \\ -1*5 + \frac{1}{3}*2 + 2/3*3 \\ -1*5 + 2/3*2 + 1/3*3 \end{bmatrix}$

$\beta = (X^T X)^{(-1)} X^T y = \begin{bmatrix} (90 - 16 - 21)/3 \\ (-15 + 2 + 6)/3 \\ (-15 + 4 + 3)/3 \end{bmatrix}$

$\beta = \begin{bmatrix} 53/3 \\ -7/3 \\ -8/3 \end{bmatrix}$

The model is $y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$ $\quad y = \frac{53}{3} - \frac{7}{3}x_1 - \frac{8}{3}x_2$

Class Activity- Solution (2)

- Predict the value of y for [4 5] and [3 4]

$$y = \frac{53}{3} - \frac{7}{3}x_1 - \frac{8}{3}x_2$$

For [4 5], $y = \frac{53}{3} - \frac{7}{3} * 4 - \frac{8}{3} * 5$

For [4 5], y =5

For [3 4], $y = \frac{53}{3} - \frac{7}{3} * 3 - \frac{8}{3} * 4$

For [3 4], y = 0

Comment on Problem (1)

ALTERNATE TECHNIQUE

$5 = b_0 + b_1 * 2$
$2 = b_0 + b_1 * 1$

Solving
$5 = b_0 + b_1 * 2$
$4 = 2 * b_0 + b_1 * 2$

$b_0 = -1$
$b_1 = (5 - b_0)/2 = (5 - (-1))/2 = 6/2 = 3$
$y = -1 + 3 * x_1$

ALTERNATE TECHNIQUE MAY NOT ALWAYS WORK!

Feature1 (x_1)	y
2	5
1	2
3	7

Class Activity-Solution to (1)

ALTERNATE TECHNIQUE MAY NOT ALWAYS WORK!

Feature1 (x_1)	y
2	5
1	2
3	7

$$X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 1 & 3 \end{bmatrix} \qquad X^T = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \end{bmatrix}$$

$$X^T.\ X = \begin{bmatrix} 1 & 2 \\ 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 1*1+2*2 & 1*1+2*1 & 1*1+2*3 \\ 1*1+1*2 & 1*1+1*1 & 1*1+1*3 \\ 1*1+3*2 & 1*1+3*1 & 1*1+3*3 \end{bmatrix} = \begin{bmatrix} 5 & 3 & 7 \\ 3 & 2 & 4 \\ 7 & 4 & 10 \end{bmatrix}$$

END
SESSION

Table of Contents

Classification Metrics

- R^2 and RMSE do not apply to classification as the difference between predicted class and actual class is undefined and meaningless.

- Classification models usually generate two types of predictions. a predicted class which comes in the form of a discrete category.

For most practical applications a discrete category prediction is required in order to make a decision. For example is email a spam we need yes or no answers. However the model would classify as spam an email that .51 likely to be spam as spam and so the same for .99.

Classification Metrics

- classification models produce a continuous valued prediction CVP which is usually in the form of a probability (i.e. the predicted values of class membership for any individual sample are between 0 and 1 and sum to 1. For example setosa verginica or versicolor could be [.2 .6 .2]
- There are some algorithms that give CVP that do not add to 1 for example we could have [.6 .5 .1] In this case we use *softmax* function to convert $\hat{y}_l$ it to probability

$$\hat{p}_\ell^* = \frac{e^{\hat{y}_\ell}}{\sum_{l=1}^{C} e^{\hat{y}_l}}$$

where $\hat{y}_l$ is the numeric model prediction for the l^{th} class and $\hat{p}_l$ is the transformed value between 0 and 1.

- Suppose that an outcome has three classes and that a model predicts values of $\hat{y}_l = 0.25$ $\hat{y}_2 = 0.76$ and $\hat{y}_3 = -0.1$.
The softmax function would transform these values to $\hat{p}_l = 0.30$ $\hat{p}_2 = 0.49$ and $\hat{p3} = 0.21$.

Binary Classification Metrics –Accuracy

The simplest metric is the overall ***accuracy*** rate (or for pessimists the error rate). This reflects the agreement between the observed and predicted classes and has the most straightforward interpretation. First overall accuracy counts make no distinction about the *type* of errors being made.

For example if among 15 cats 10 cats were identified correctly and among 15 dogs 8 dogs were identified correctly the accuracy is 18/30 = 60%

Binary Classification Metric - Confusion Matrix

The ***confusion matrix*** for the two-class problem ("events" and "nonevents." The table cells indicate number of the true positives (*TP*) false positives (*FP*) true negatives (*TN*) and false negatives (*FN*)

Predicted	Observed	
	Event	Nonevent
Event	*TP*	*FP*
Nonevent	*FN*	*TN*

Example:

Predicted	Observed	
	Cats	Dogs
Cats	*10*	*7*
Dogs	*5*	*8*

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$$

Accuracy = 18/30 = 60%

Binary Classification Metric – Precision(PPV)

$$\text{Precision} = \frac{TP}{TP+FP}$$

Precision is also called ***Positive Predictive Value (PPV)***. ***How reliable is the positive result of the model?*** It is used whenever we want to *avoid false positives*. For example when a new drug is developed we can check its effectives with positive if it is effective negative if It is not. We want to make sure it is most effective that is it has least false positives. False negatives do not affect the patients.

For the Cats versus Dogs example Precision = 10/17 Among the 17 samples predicted as CATs 10 were correct! That is 58% of the of time the model found cats it was correct.. Not too good if cats are of your primary interest! Note that overall accuracy was 60%

Binary Classification Metric – Recall (Sensitivity)(TPR)

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

Recall is also called *sensitivity hit rate* or *true positive rate* (TPR). **How good is the model at catching all positive results?** Recall is used as performance metric when we need to identify all positive samples; that is when it is important to *avoid false negatives*. Consider cancer diagnosis as an example. It is important to find all people that are sick possibly including healthy patients in the prediction.

For the Cats versus Dogs example Recall = 10/15 Among the 15 samples predicted as CATs model could identify only 10 of them. That is 66% of the of time the model found cats among all cats. The hit rate was therefore 66%

Binary Classification Metric – Specificity(TNR)

$$\text{Specificity} = \frac{TN}{TN + FP}$$

Specificity also known as True Negative Rate refers to answering the question **How good is the model at catching all negative results?** It is the proportion of those that got correct negative results among the ones that should have gotten negative results.

Binary Classification Metric – f-statistic

- There is a trade-off between precision and recall.
- You can get 100% recall if you predict all samples as positive.
- You can get 100% precision if you predict all samples as negative
- So f-measure captures the combination. Highest value can be 1 and least can be 0. It gives a better measure of accuracy of the model.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

For the Cats versus Dogs example the F-score

F = 2 * (10/17 *10/15)/(10/17 + 10/15) = 2 * .392/1.254 = 0.625 = 62.5%

Computation

```
from sklearn.metrics import confusion_matrix
confusion = confusion_matrix(y_test, pred_logreg)
print("Confusion matrix:\n{}".format(confusion))
```

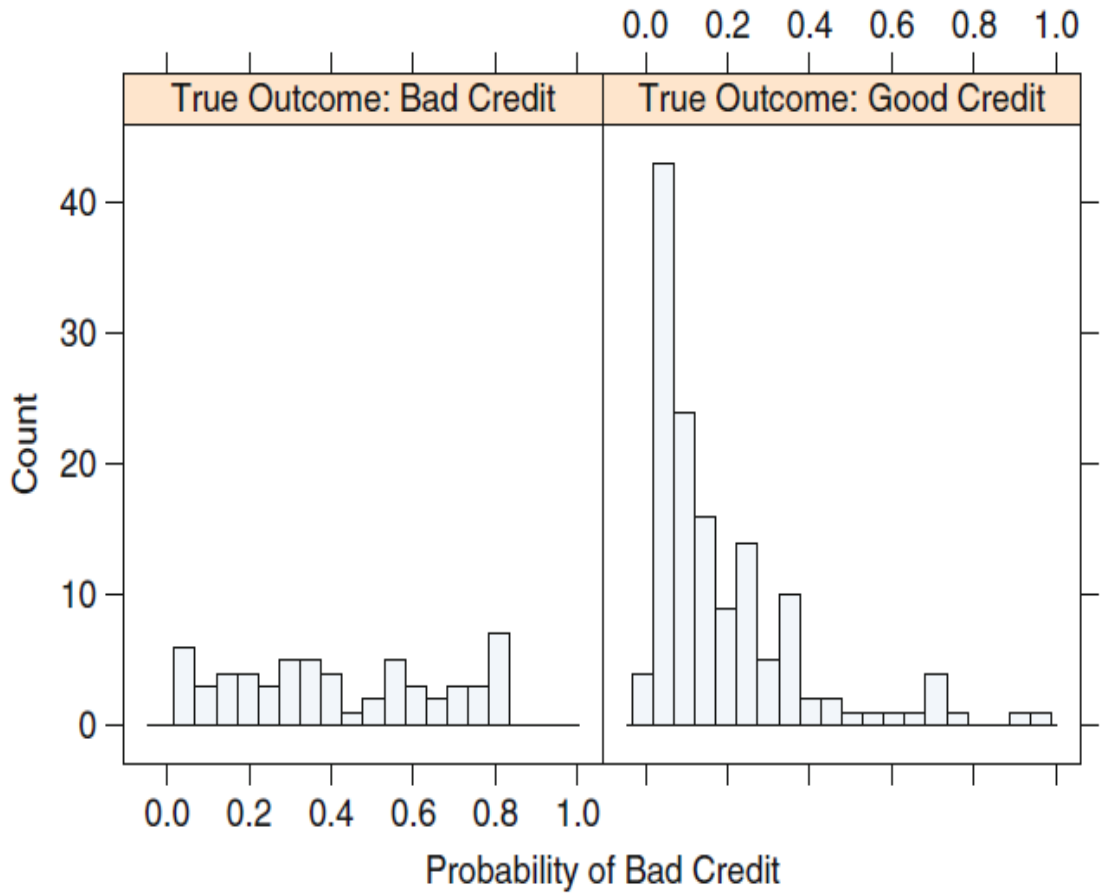
```
Confusion matrix:
[[401 2]
[ 8 39]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred_most_frequent,
target_names=["not nine", "nine"]))
```

```
precision recall f1-score support
not nine                0.90     1.00   0.94     403 (number of samples of not nine)
nine                    0.00     0.00   0.00     47 (number of samples of nine)
```

Binary Classification Metric – Histogram

For customers with Bad Credit the count of samples with predicted probability

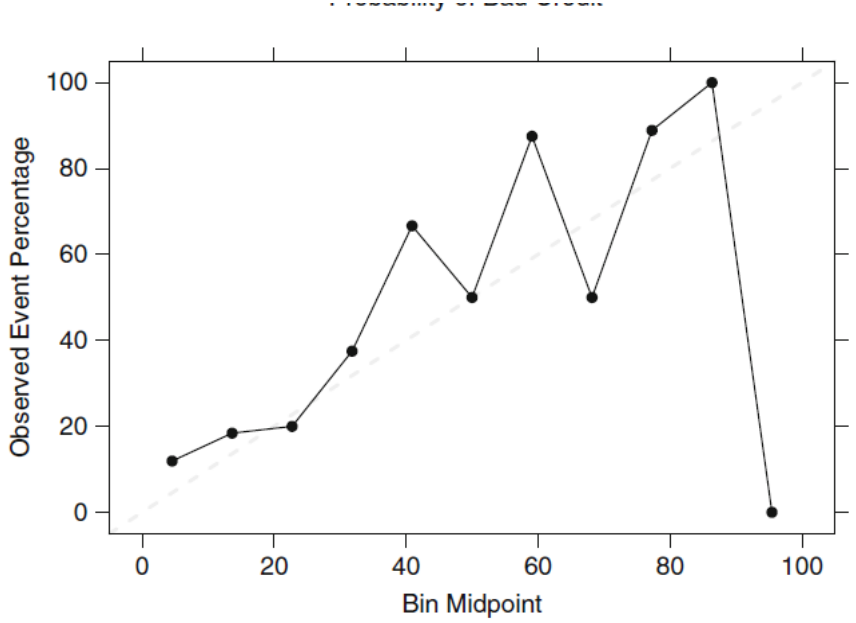


For customers with Good Credit the count of samples with predicted probability

The probability of bad credit for the customers with good credit shows a skewed distribution where most customers' probabilities are quite low. In contrast the probabilities for the customers with bad credit are flat (or uniformly distributed) reflecting the model's inability to distinguish bad credit cases.

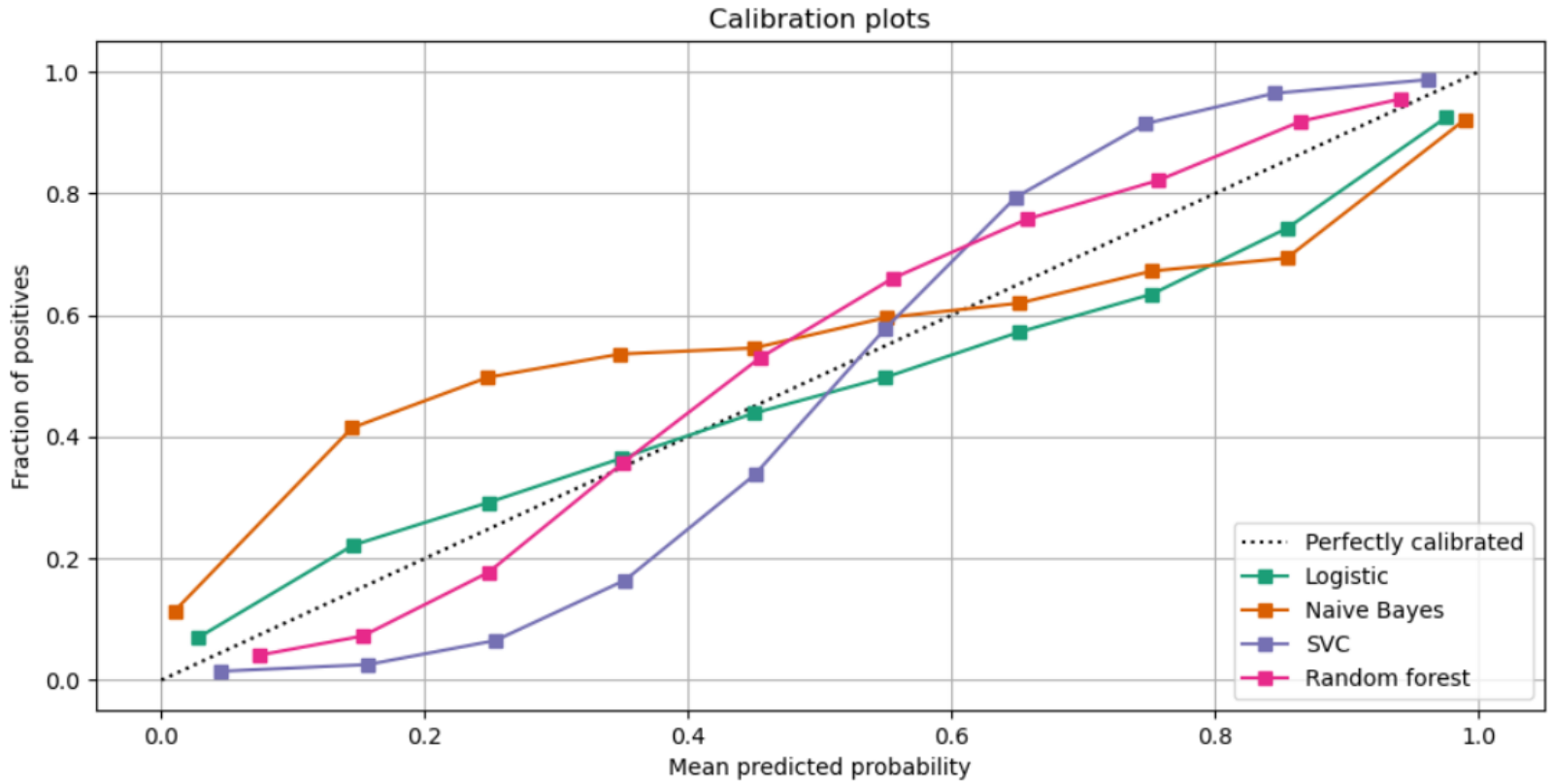
Binary Classification Metric-Calibration Plot

- a set of bins chosen might be [0 10%] (10% 20%] . . . (90% 100%]
- For each bin determine the ***observed event rate***. Per prediction, let us say 50 samples fell into the bin for class probabilities less than 10% for class A and there was a single observed event of class A among the 50 samples.
- The midpoint of the bin is 5% and the observed event rate would be 2%.(1 out of 50 so 2 out of 100)
- The calibration plot would display the midpoint of the bin on the *x*-axis and the observed event rate on the *y*-axis. If the points fall along a 45◦ line the model has produced well-calibrated probabilities.



Calibration Plot

```
sklearn.calibration.calibration_curve(y_true, y_prob, *, normalize=False,
n_bins=5, strategy='uniform')
```



Binary Classification Metric -ROC – Receiver Operating Characteristic

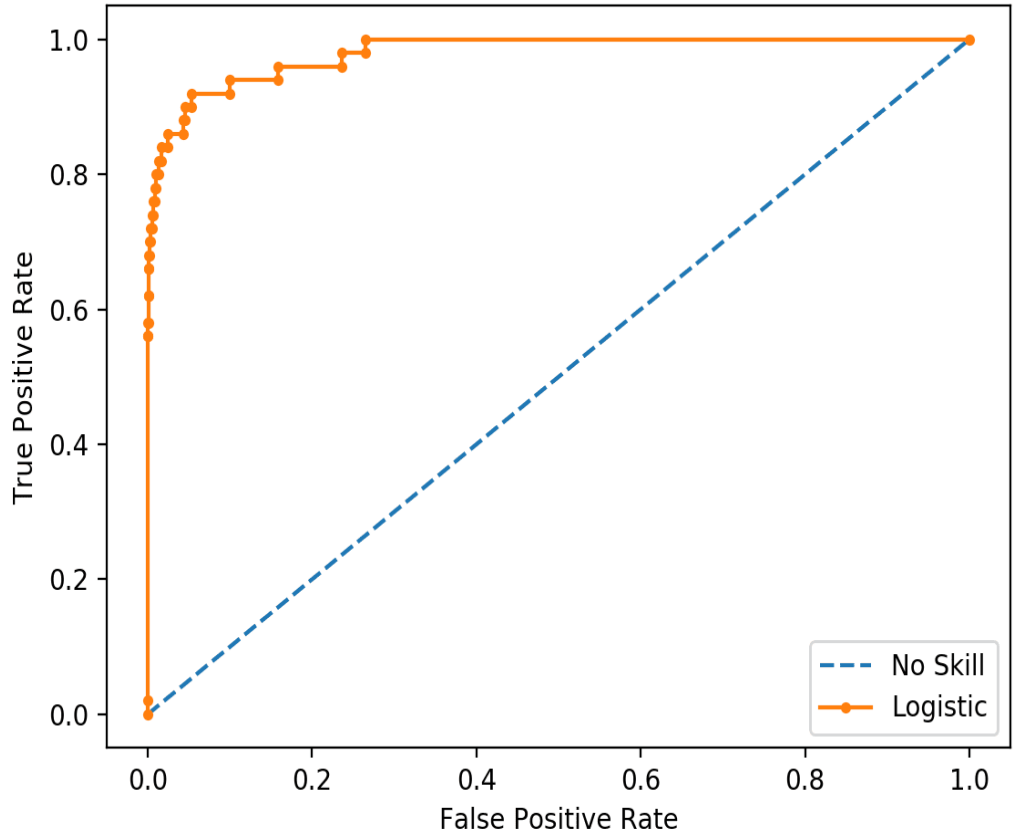
- Generally a threshold such as 0.5 is used where all values equal or greater than the threshold are mapped to one class and all other values are mapped to another class.

- For those classification problems that have a severe class imbalance the default threshold can result in poor performance.

- ROC is a technique used to find the “right threshold”. Different thresholds such as .1, .2, .3 etc can be tried and for each threshold one can find the TPR and FPR. (true positive rate and false positive rate.

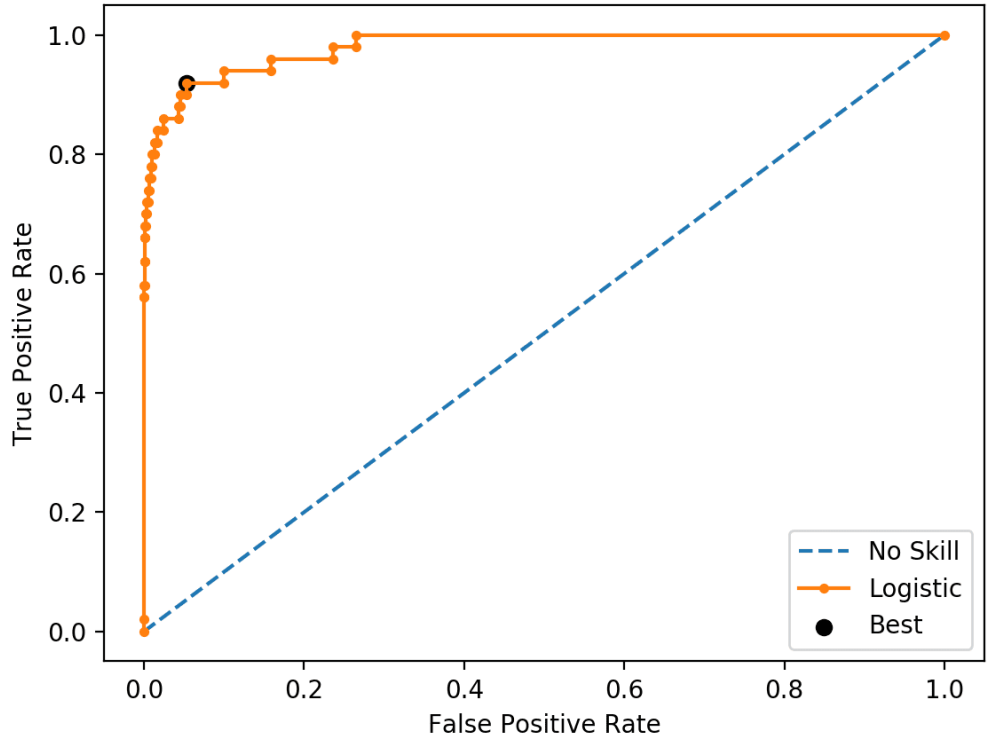
Binary Classification Metric ROC – Receiver Operating Characteristic

ROC curve shows for ***each threshold what is the FPR versus TPR***. Of course we would like FPR to be close to zero and TPR to be close to 1. **Threshold is not shown in the picture**



ROC-Curve for Threshold Determination

The program would return values as follows: For each threshold from 0 to 1 we compute FPR and TPR and return a list of the form [ThresholdList FPRList TPRList]. Then the list is examined to see the threshold where FPR is lowest and TPR is highest



ROC Curve- Threshold Determination

- The Geometric Mean or G-Mean is a metric for imbalanced classification that if optimized will seek a balance between the sensitivity and the specificity.

  G-Mean = sqrt(Sensitivity * Specificity)

- One approach would be to test the model with each threshold returned from the call to ROC_curve and select the threshold with the largest G-Mean value.

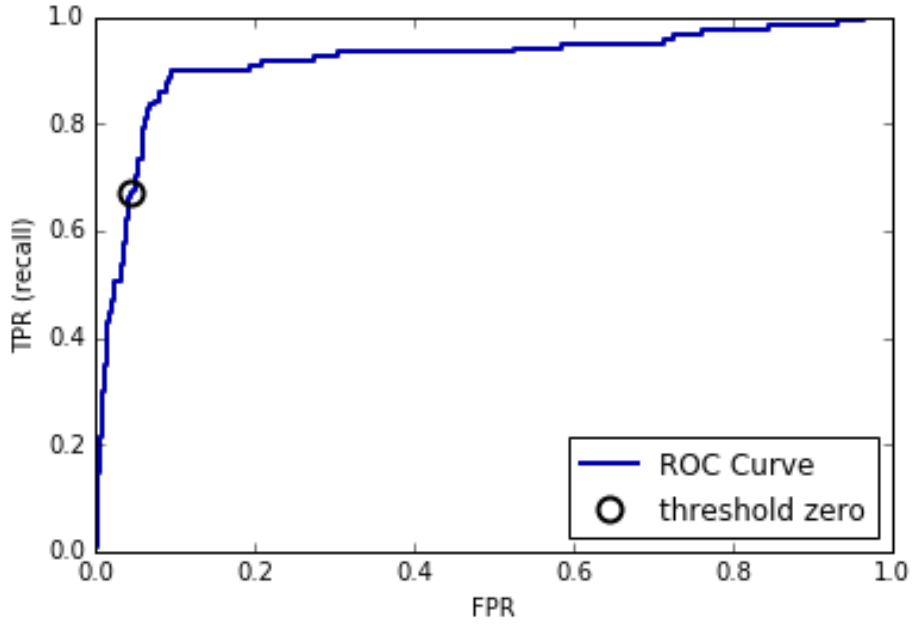
Computation

```
from sklearn.metrics import roc_curve

#The following function runs the algorithm on X_test and determines
#y_pred and compares with y_test for each threshold from 0.1 to 1.0 and
#then returns the fpr(s), tpr(s) and the threshold values.

#Then best threshold is found the G-mean computation for each set of tpr
#and fpr values

[fpr tpr thresholds] = roc_curve(y_test svc.decision_function(X_test))
```



Linear Models for Classification-Logistic Regression

Logistic Regression (is a *classification* model .. There is a misnomer!)

- The formula looks very similar to the one for linear regression but instead of just returning the weighted sum of the features we expect $\hat{y}$ to be 1 when the sample belongs to class 1 or $\hat{y}$ to be zero when the sample belongs to class 0.

$$\hat{y} = \beta_0 + \beta_1 * x_1 + \ldots + \beta_p * x_P$$

Since RHS will give us a real number which we cannot interpret as probability for a sample belonging to a class.
So, we change the LHS of the model to represent the odds of event occurring.

Linear Models for Classification-Logistic Regression

If `p` represents the probability of an event (class) then $\frac{p}{(1-p)}$ represents the odds that the data belongs to a class.

Note that $\frac{p}{(1-p)}$ is still between -inf to inf.

We use $\log\frac{p}{(1-p)}$ to model (why?) This is called ***log odds***.

We can compute `p` as

```
p = number of samples of that class/(total number of samples).
```

Logistic Regression

Our model is

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 * x_1 + \ldots + \beta_p * x_P$$

$$\frac{p}{1-p} = e^{(\beta_0 + \beta_1 * x[1] + \ldots + \beta_p * x[p])}$$

$$p = e^{(\beta_0 + \beta_1 * x_1 + \ldots + \beta_p * xP)} - p * e^{(\beta_0 + \beta_1 * x_1 + \ldots + \beta_p * xP)}$$

$$p * (1 + e^{(\beta_0 + \beta_1 * x_1 + \ldots + \beta_p * xP)}) = e^{(\beta_0 + \beta_1 * x_1 + \ldots + \beta_p * xP)}$$

$$p = \frac{e^{(\beta_0 + \beta_1 * x_1 + \ldots + \beta_p * xP)}}{(1 + e^{(\beta_0 + \beta_1 * x_1 + \ldots + \beta_p * xP)})} * \frac{e^{-(\beta_0 + \beta_1 * x_1 + \ldots + \beta_p * xP)}}{e^{-(\beta_0 + \beta_1 * x_1 + \ldots + \beta_p * xP)}}$$

$$p = \frac{1}{1 + \exp\left[-(\beta_0 + \beta_1 x_1 + \cdots + \beta_P x_P)\right]} \qquad (12.3)$$

Logistic Regression

Our model is

$$\log\frac{p}{(1-p)} = \beta_0 + \beta_1 * x_1 + \ldots\ldots + \beta_P * x_P$$

where p is computed as probability that the sample belongs to a class. We would like to find β's such that error between predicted value and training set outcome is minimized.

Once we find the β that gives minimum error we will use the function below to make decision on class the sample belongs to. Note that p is between 0 to 1.

$$p = \frac{1}{1 + \exp\left[-(\beta_0 + \beta_1 x_1 + \cdots + \beta_P x_P)\right]} \qquad (12.3)$$

Logistic Regression

- In linear regression we minimized the following MSE (Mean square error) to find the optimal model parameters β^s . However, in logistic regression, y_i and $\hat{y}_i$ are 0 or 1.

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

- So each term will be 0 or 1, so MSE is meaningless.
- So what do we minimize for the optimization of β^s

Logistic Regression

- For all the samples that are in class 1 i.e $y_i=1$ we will predict with probability p_i that it is 1 and with probability $(1-p_i)$ that it is in class 0. ($y_i=0$). So for all samples where $y_i=1$ we would like that

$$\prod_{(y_i=1)} p_i$$ to be as close to 1 as possible.

- Similarly for all samples with $y_i=0$ $(1-y_i=1)$ we will predict with probability $(1-p_i)$ that it is 0. So we would like that

$$\prod_{(y_i=0)} (1-p_i)$$ to be as close to 1 as possible.

Logistic Regression

We minimize

$$S = \prod_{(y_i=1)} p_i \prod_{(y_i=0)} (1-p_i) = \prod_{y_i} p_i^{y_i}(1-p_i)^{(1-y_i)}$$

$$\log(S) = \sum y_i * p_i + (1-y_i) * (1-p_i) \quad \text{where } p_i = 1/(1+e^{(-\beta * x_i)})$$

Numerical methods are used to find β^s that minimize the log(S).

Computation

```
NOTE: scikit applies penalty by default!
from sklearn.linear_model import LogisticRegression
X y = mglearn.datasets.make_forge()
for model ax in zip([LogisticRegression()] axes):
clf = model.fit(X y) // plot code not included
```

Penalized Models

One might include a penalty term for the logistic regression model in a manner that is very similar to ridge regression. Recall that logistic regression finds parameter values that maximizes the binomial likelihood function $L(p)$

$$\log L(p) - \lambda \sum_{j=1}^{P} \beta_j^2.$$

A simple approach to regularizing this model would be to add a squared penalty function to the log likelihood and find parameter estimates that maximize

Penalized Models

The glmnet models uses ridge and lasso penalties simultaneously like the elastic net but structures the penalty slightly differently:

$$\log L(p) - \lambda \left[(1-\alpha)\frac{1}{2}\sum_{j=1}^{P}\beta_j^2 + \alpha\sum_{j=1}^{P}|\beta_j| \right].$$

Here the *a* value is the "mixing proportion" that toggles between the pure lasso penalty (when $a = 1$) and a pure ridge-regression-like penalty ($a = 0$).The other tuning parameter λ controls the total amount of penalization.

This is called ElasticNet in scikit-learn

Linear Discriminant Analysis

- Classification is done three ways

1. - Algorithmically (k-NN)
2. - Mathematical Setup but solved iteratively (Logistic Regression)
3. - Fully Mathematically (LDA)

Linear Discriminant Analysis (Naïve Bayes)

- Mathematical technique is minimize misclassification. We need to understand Bayes Rule to understand LDA

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

A, B	= events
$P(A\|B)$	= probability of A given B is true
$P(B\|A)$	= probability of B given A is true
$P(A), P(B)$	= the independent probabilities of A and B

Linear Discriminant Analysis (Naïve Bayes)

$$Pr[Y = C_\ell | X] = \frac{Pr[Y = C_\ell] Pr[X | Y = C_\ell]}{\sum_{l=1}^{C} Pr[Y = C_l] Pr[X | Y = C_l]}$$

Pr [*Y* = C_l] is known as the *prior probability* of membership in class *C*. In practice these values are either known are determined by the proportions of samples in each class or are unknown in which case all values of the priors are set to be equal.

Pr [*X*|*Y* = C_l] is the *conditional probability* of observing predictors *X* given that the data stem from class *C*. Here we assume that the data are generated from a probability distribution (e.g. multivariate normal distribution) which then defines this quantity's mathematical form.

Pr[*Y* = C_l |*X*] which is commonly referred to as the *posterior probability* that the sample *X* is a member of class *C*.

Linear Discriminant Analysis (Naïve Bayes)

Once $Pr[Y = C_l \mid X]$ is determined how do we make the decision if it belongs to class l?

For a two-group classification problem the rule that minimizes the total probability of misclassification would be to classify X into group 1 if $Pr[Y = C1 \mid X] > Pr[Y = C2 \mid X]$ and into group 2 if the inequality is reversed. Using Equation given in previous slide this rule directly translates to classifying X into group 1 if

$$Pr[Y = C_1]Pr[X|Y = C_1] > Pr[Y = C_2]Pr[X|Y = C_2].$$

$$Pr[Y = C_\ell|X] = \frac{Pr[Y = C_\ell]Pr[X|Y = C_\ell]}{\sum_{l=1}^{C} Pr[Y = C_l]Pr[X|Y = C_l]}$$

Linear Discriminant Analysis (Naïve Bayes)

- For classification the number of predictors is almost always greater than one and can be extremely large. In more realistic situations how does one compute quantities such as $Pr[X|Y = C_l]$ in many dimensions? What multivariate probability distributions can be used to this effect?

$$X'\Sigma^{-1}\mu_\ell - 0.5\mu_\ell'\Sigma^{-1}\mu_\ell + \log\left(Pr[Y = C_\ell]\right).$$

One special often used scenario is to assume that the distribution of the predictors is multivariate normal. This distribution has two parameters: the multidimensional mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Further if we assume that the means of the groups are unique (i.e. a different $\boldsymbol{\mu}_l$ for each group) but the covariance matrices are identical across groups we can solve the more general multi-class problem to find the linear discriminant function of the l^{th} group:

Computation

```
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
 X = np.array([[-1 -1] [-2 -1] [-3 -2] [1 1] [2 1] [3 2]])
 y = np.array([1 1 1 2 2 2])
clf = LinearDiscriminantAnalysis()
clf.fit(X y)
clf.predict([[-0.8 -1]])
```

Class Work

1) Given the following results of binary classification problem, find the Accuracy, Positive Predictive Rate (Precision), True Positive Rate (sensitivity, Recall), True Negative Rate(specificity) and f-statistic

	Observed Results	
	Movie Worth Watching	Not worth watching
	15	6
Predicted Results	4	8

Class Work Solution (1)

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{Specificity} = \frac{TN}{TN+FP}$$

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision}+\text{recall}}$$

Predicted Results	Observed Results: Movie Worth Watching	Not	
Worth	15(TP)	6(FP)	21
Not	8 (FN)	4 (TN)	12
	23	10	

Accuracy = (15+ 4)/33 = 19/33 = .575
Precision = 15/21=.71
Recall = 15/23=.65
Specificity = 4/10=.4

F = 2 * (.65 *.71)/(1.36) = .67

Class Work

2) Given the following classification data

a) Find the $Pr\,[Y = C_j]$

b) find the log odds values for each sample.

Description of Features

1.Number of times pregnant. 2. Plasma glucose concentration in a 2 hours in an oral glucose tolerance test. 3. Diastolic blood pressure (mm Hg). 4. Triceps skinfold thickness (mm). 5. 2-Hour serum insulin (mu U/ml). 6. Body mass index (weight in kg/(height in m)^2). 7. Diabetes pedigree function. 8. Age (years). 9. Class variable (0 or 1).

1	2	3	4	5	6	7	8	9
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1

Class Work Solution (2)

1	2	3	4	5	6	7	8	9	p/1-p	log(p/1-p)
6	148	72	35	0	33.6	0.627	50	1	.6/.4 =1.5	0.176
1	85	66	29	0	26.6	0.351	31	0	.4/.6 = 0.6	0.22
8	183	64	0	0	23.3	0.672	32	1	.6/.4 = 1.5	0.176
1	89	66	23	94	28.1	0.167	21	0	.4/.6 = 0.6	0.22
0	137	40	35	168	43.1	2.288	33	1	.6/.4 = 1.5	0.176

a) Prob(y=1) = 3/5 = 0.6, and Prob(y = 0) = 2/5 = 0.4