Introduction to Statistical Learning

Definitions

- Suppose we are given two variables X and Y, represented by a sample of n points of the form (x_i; y_i), for 1<= i <= n.
- X and Y are **correlated** when the value of X has some predictive power on the value of Y.
- Correlation coefficient **r(X; Y)**
  - Measures the degree to which Y is a function of X, and vice versa.
  - Its value ranges from -1 to 1
  - 1 means fully correlated and 0 => n relation, i.e. independence
  - -ve => variables are anti-correlated, if X goes up, Y goes down.

The Pearson Correlation Coefficient

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} = \frac{Cov(X,Y)}{\sigma(X)\sigma(Y)}$$

- The elements of numerator are +ve when samples of both X and Y are greater than the mean or smaller at the same time. This would mean that they are positively correlated.
- If X is usually greater than the mean when Y is less than its mean, i.e. the signs are –ve, this would mean they are –vely correlated.
- Works well for linear predictors (Y = aX + b)

Detect defective measurements

- Traffic flow measurement conducted in 4 different lanes of a highway.
- The measurement device develops defects and takes incorrect readings
- This can be detected by using the measurements in other lanes.

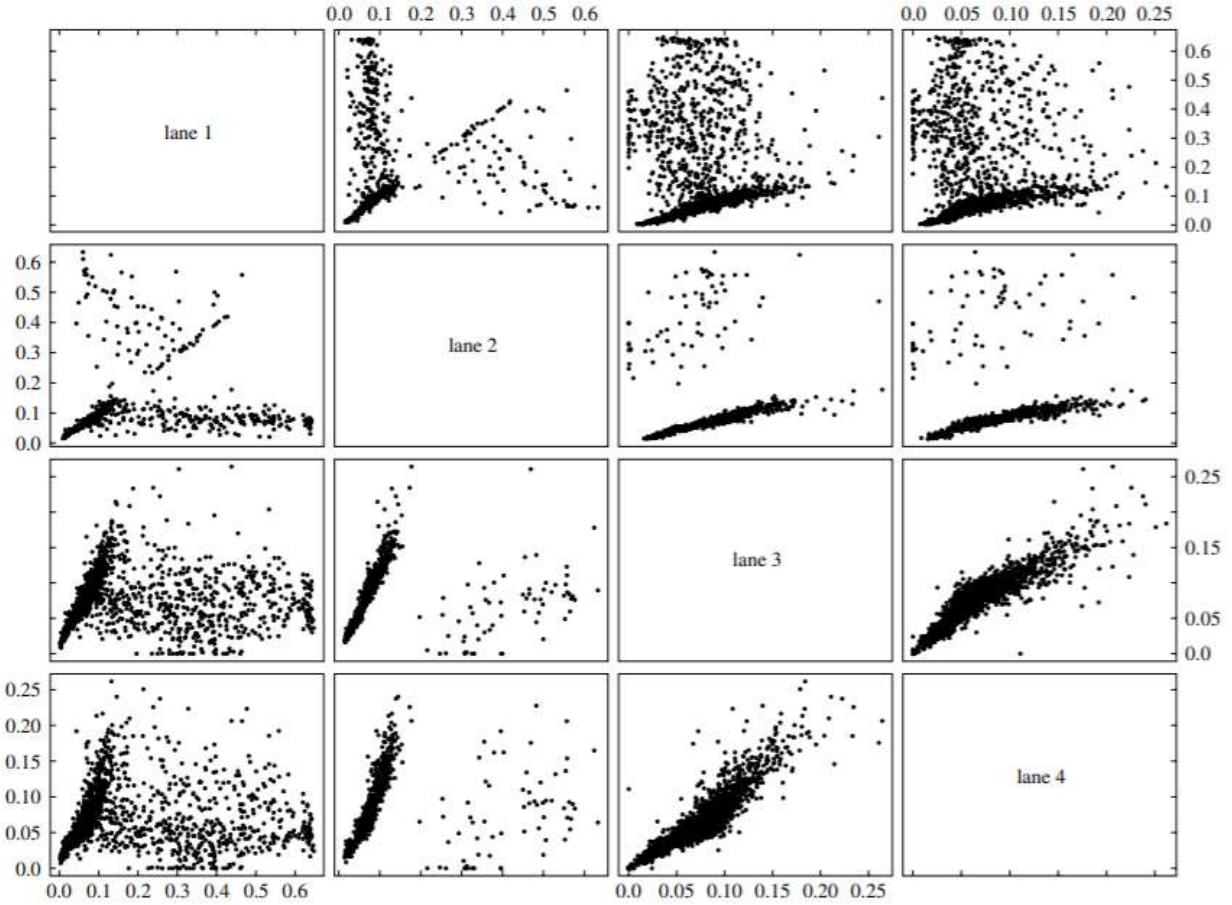


FIGURE **10.17** Occupancy measurements by adjacent loops in four lanes.

The Spearman Rank Correlation Coefficient

- Count the number of pairs of input points which are out of order.
- Suppose the data set contains points (x1; y1) and (x2; y2) where x1 < x2 and y1 < y2.
- This is a vote that the values are positively correlated, whereas the vote would be for a negative correlation if y2 < y1.
- Summing up over all pairs of points and normalizing properly gives us Spearman rank correlation.
- This method is much more robust against outliers

The Spearman Rank Correlation Coefficient

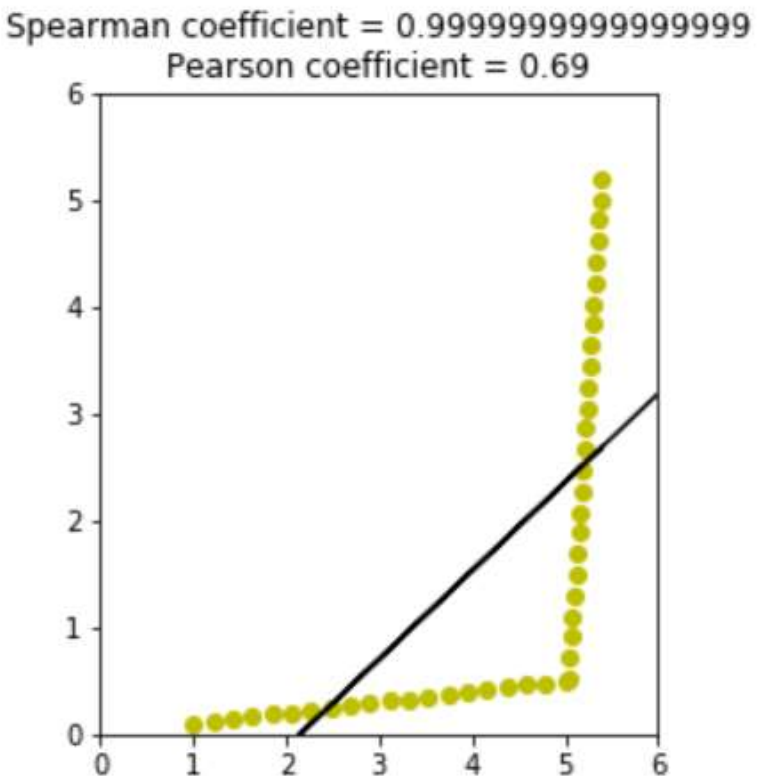
- Let rank(x_i) be the rank position of x_i in sorted order among all x_i, so the rank of the smallest value is 1 and the largest value n.
- Then spearman rank correlation coefficient is,

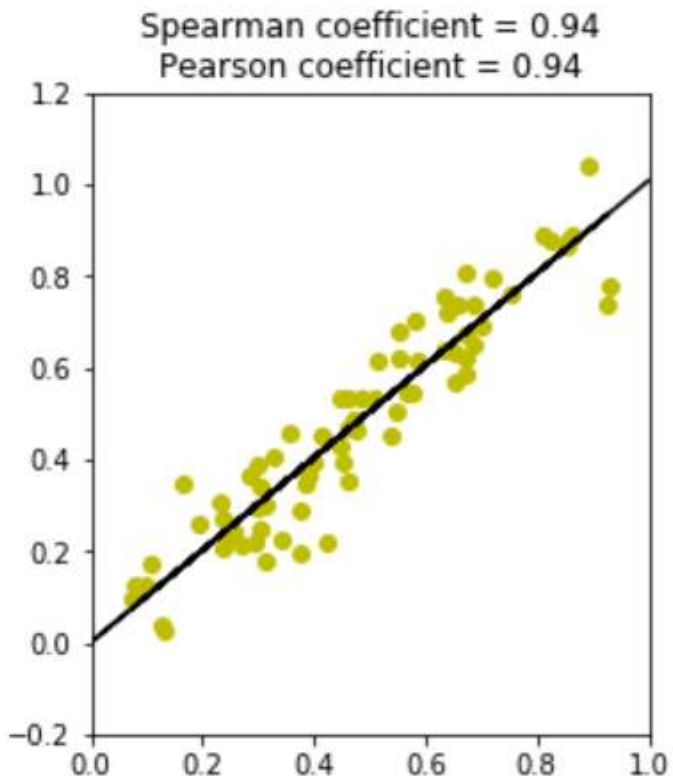
$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

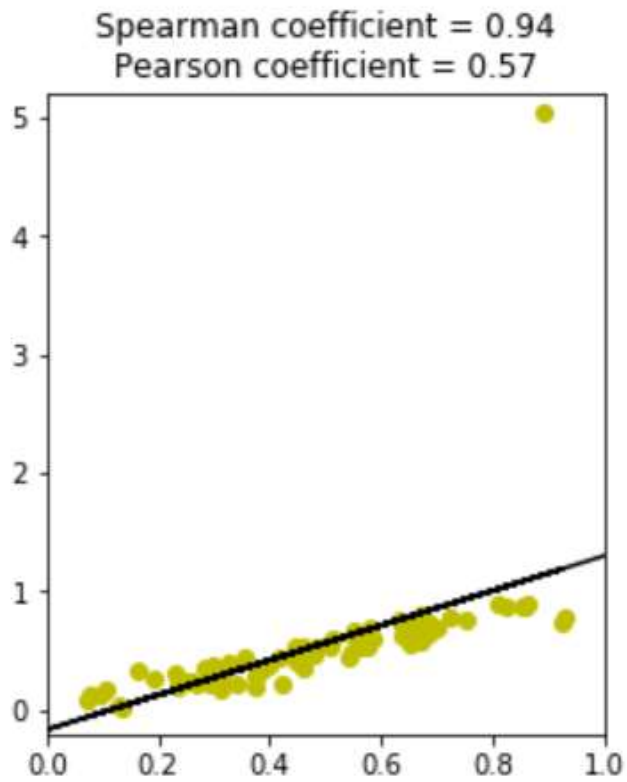
$$\text{where } d_i = rank(x_i) - rank(y_i).$$

Exercise: Prove that $-1 < \rho < 1$ for any value of n

Comparison







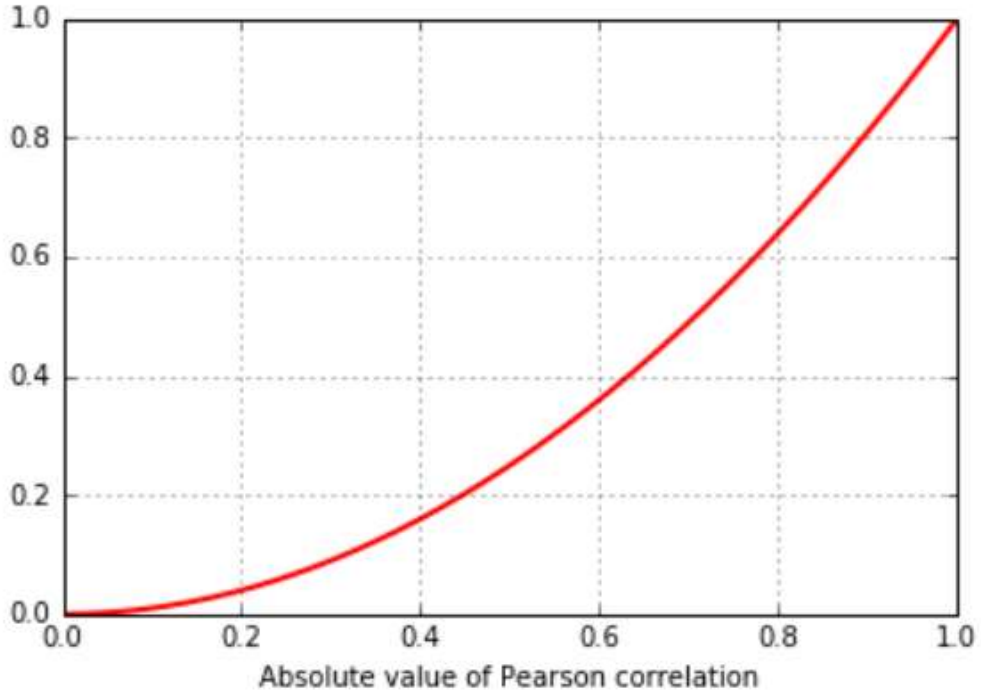

```
rho, pval = stats.spearmanr(xval, yval)
pearson = stats.pearsonr(xval, yval)
```

True or False?

- Taller people are more likely to remain lean
- Medicine causes illness
- Standardized tests like GRE, Board exams predict the performance of students in college/university
- Financial status affects health
- Smoking causes cancer
- Active Police in an area increases crime
- Wealth decreases family size

Strength of Correlation

- R^2 : The strength of the correlation is defined as the square of the sample correlation coefficient. It estimates the fraction of variance in Y explained by X in a **simple linear regression**.



Statistical significance

- The statistical significance of a correlation depends upon its sample size *n* as well as *r*.
- A correlation of n points is significant if there is an $\alpha \leq \frac{1}{20} =$ 0.05 chance that we would observe a correlation as strong as *r* in any random set of *n* points drawn from the population.

Statistical Models

Response and Predictor Variables

- We are observing $p + 1$ numerical variables and we are making n sets of observations.
- We call the variable we'd like to predict the **outcome** or **response variable**; typically, we denote this variable by Y and the individual measurements y_i.
- The variables we use in making the predictions the **features** or **predictor** variables; typically, we denote these variables by $X = X_1, \ldots, X_p$ and the individual measurements $x_{i,j}$.

Note: i indexes the observation ($i = 1, \ldots, n$) and j indexes the value of the j-th predictor variable (j $= 1, \ldots, p$). Total number of predictor variables, J=p

True vs. Statistical Model

- We will assume that the response variable, Y, relates to the predictors, X, through some unknown function 'f' expressed generally as:

$$Y = f(X) + \varepsilon$$

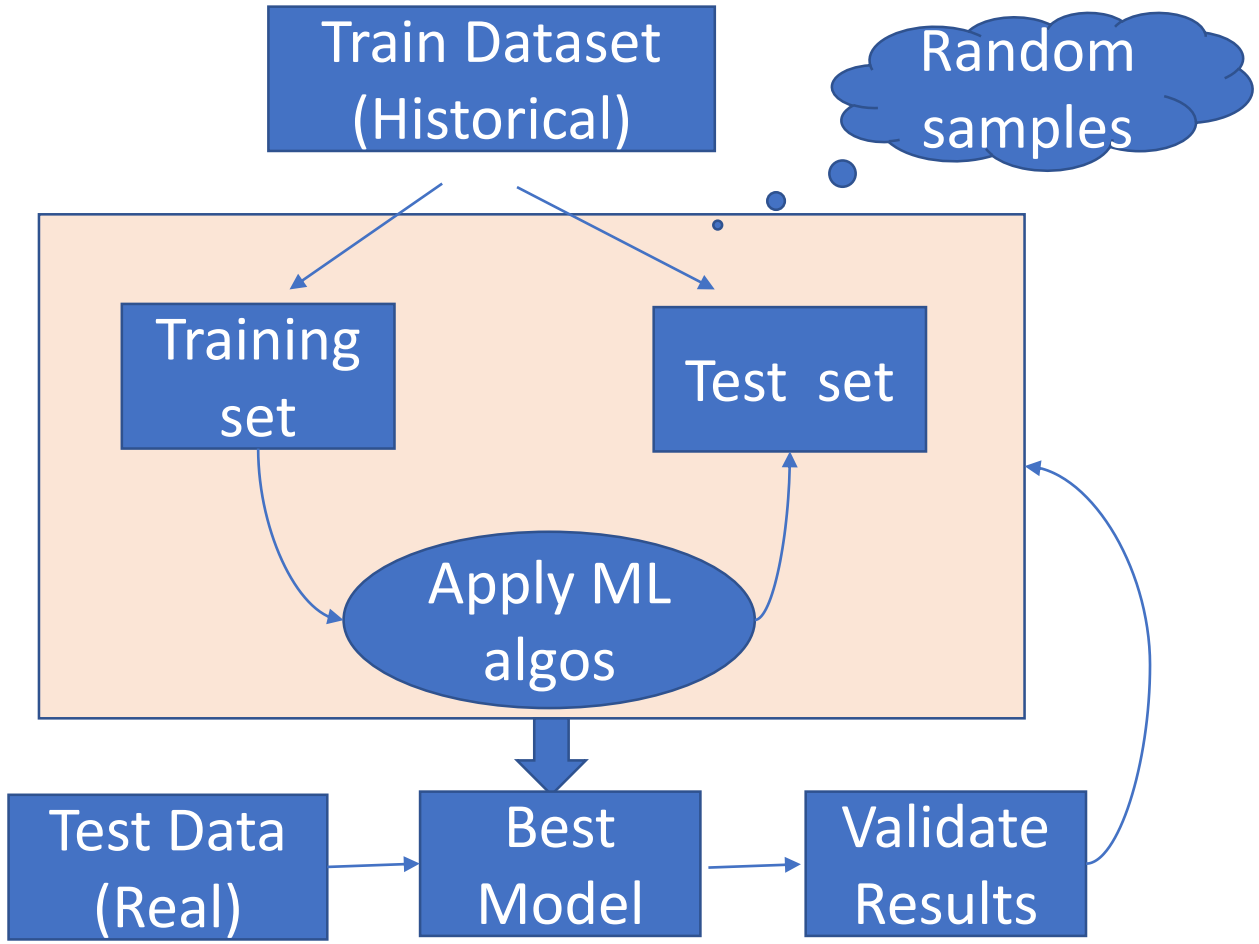
- Here, f is the unknown function expressing an underlying rule for relating Y to X, ε is the random amount (unrelated to X) that Y differs from the rule $f(X)$.

- A ***statistical model*** is any algorithm that estimates f. We denote the estimated function as $\widehat{f}$.

Machine Learning

- **ML algorithms have the objective of generalization**, i.e. use one dataset to generate models that perform well on data that they have not seen.
- Thus, they prove to be effective in generating predictive models.
- There are many ML techniques and ML models have several parameters
- How to choose the best one?

Machine Learning Methodology



- Input dataset is divided into a random split (80/20) or (90/10) to be used for training and testing respectively
- For each split, model is generated and tested for accuracy
- This is repeated 5 times or 10 times and average error is computed
- The best model is selected and used in real world scenario

Goals of Machine Learning

- Prediction: We are only interested in the correct output “y”. The model may be black-box, we may not know the relationships between x and y and which attributes influence y in what way etc.
- Inference: We are more interested in the relationships of the attributes with “y”. We want to know which attributes are more important, how they influence “y”.
- Inference + Prediction: We are interested in both.

Flexibility vs. Interpretability Tradeoff

- There are many methods of regression (that estimate f)
- Some are less flexible but more interpretable
- These are useful for inference problems where we want to study the relationships between predictor variables
- But highly flexible methods can also lead to over-fitting!

Error Evaluation

In order to quantify how well a model performs, we define a ***loss*** or ***error function.***

A common loss function for quantitative outcomes is the **Mean Squared Error (MSE):**

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

The quantity $y_i - \hat{y}_i$ is called a ***residual*** and measures the error at the *i*-th prediction.

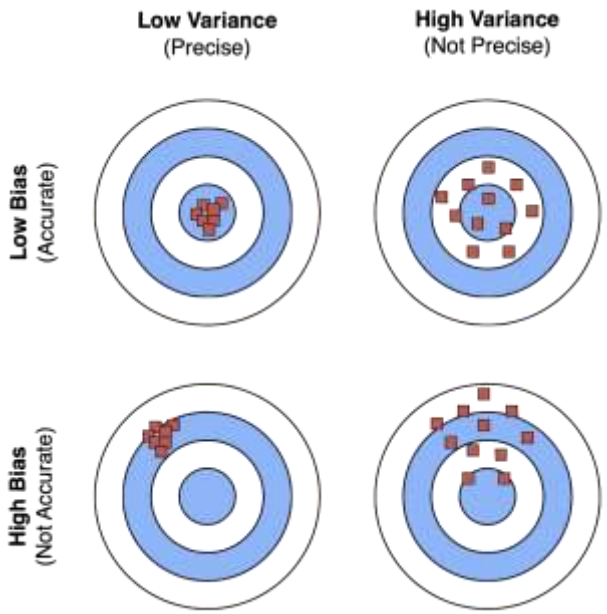
The square root of MSE is RMSE: $RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2}$

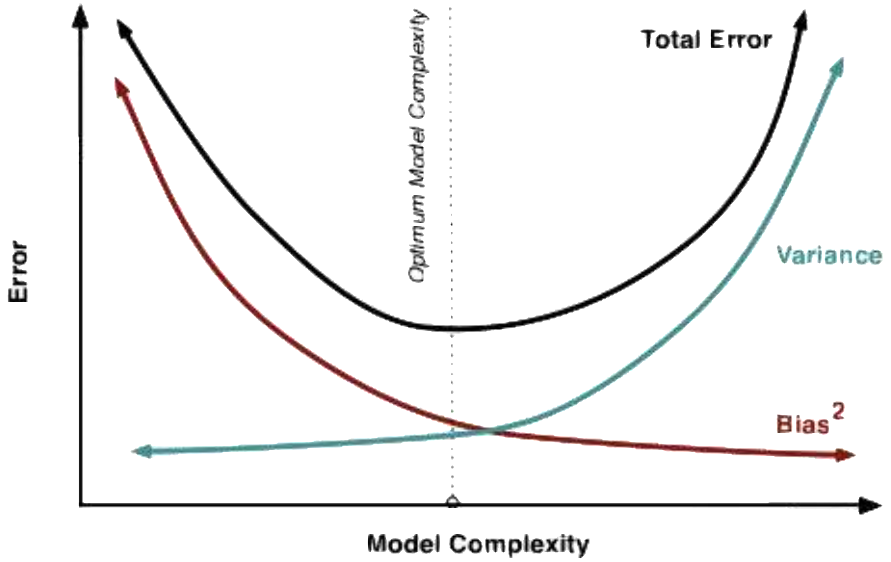
R-squared Error

$$R^2 = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (\bar{y} - y_i)^2}$$

- If our model is as good as the mean value, $\bar{y}$, then $R^2 = 0$
- If our model is perfect then $R^2 = 1$
- R^2 can be negative if the model is worst than the average. This can happen when we evaluate the model in the real life test set.

Bias Variance Tradeoff





- Total Error = Bias^2 + Variance + Irreducible Error

Bias is the average distance of estimate $\hat{f}$ from the true mean of f(x)

Variance is the sq. dev of the estimate $\hat{f}$ around its mean

Bias Variance Tradeoff

"All models are wrong, but some models are useful." : George Box (1919-2013)

- Occam's razor: This philosophical principle states that "the simplest explanation is best".
- ***Bias*** is error from erroneous assumptions in the model, like making it linear/simplistic. (underfitting)
- ***Variance*** is error from sensitivity to small fluctuations in the training set, indicating it will not work in real world. (overfitting)
- First-principle models likely to suffer from bias, with data-driven models in greater danger of overfitting.

Example Problem (Advertising)

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

The **Advertising data set** consists of the sales of that product in 200 different markets, along with advertising budgets for the product in each of those markets for three different media: TV, radio, and newspaper. Everything is given in units of $1000.

Some of the figures in this presentation are taken from ISL book: "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani "

Response vs. Predictor Variables

X
predictors
features
covariates

Y
outcome
response variable
dependent variable

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9

n observations

p predictors

k-Nearest Neighbors

The ***k-Nearest Neighbor (kNN) model*** is an intuitive way to predict a quantitative response variable:

> *to predict a response for a set of observed predictor values, we use the responses of other observations most similar to it*

Note: this strategy can also be applied in classification to predict a categorical variable.

k-Nearest Neighbors

For a fixed a value of *k*, the predicted response for the i-th observation as the average of the observed response of the *k*-closest observations:

$$\hat{y}_n = \frac{1}{k} \sum_{i=1}^{k} y_{n_i}$$

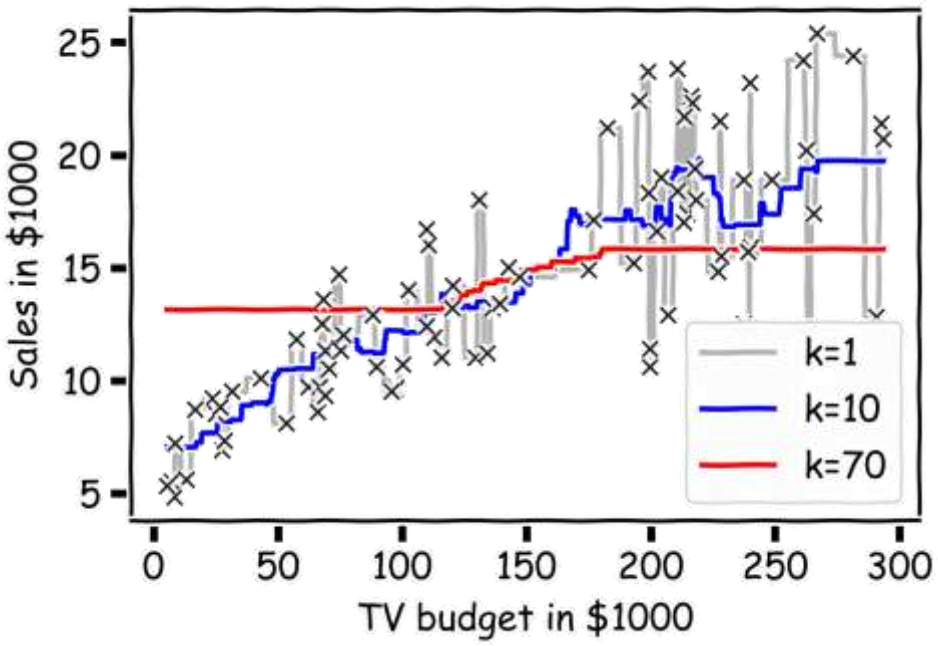
where $\{x_{n1}, \ldots, x_{nk}\}$ are the *k* observations most similar to x_i (*similar* refers to a notion of distance between predictors).
Usually, Euclidean distance is chosen (sqrt of squared coordinate differences)

Python: sklearn.neighbors.KNeighborsRegressor(n_neighors=3)

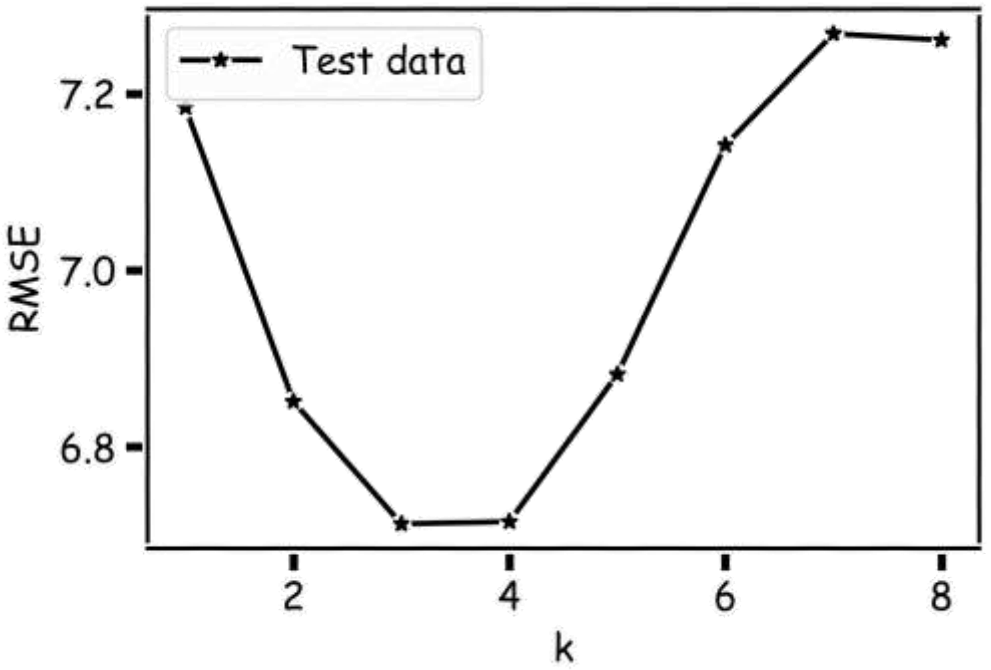
4-Nearest Neighbors

- Very few assumptions made here about the nature of ‘f’
- Equal weights for the values of y, regardless of their distance from x
- As dimensionality increases, becomes hard to find neighbors closeby and f may change significantly

Model Comparison



Q. Reason for discontinuity
Q. Bias Variance tradeoff



Do the same for all *k*'s and compare the RMSEs
Which k is best?

Linear Models

- In the kNN approach, we didn't assume a form of the function 'f'
  - Such approaches are called non-parametric approaches
- In the linear regression approach, we assume that the response is a linear function of the predictor variables
- Note that this technique can be easily extended by creating extra predictor variables (features) from a combination (transformation) of the original predictor variables.
- So lets assume:

$$Y = f(X) + \epsilon = \beta_1 X + \beta_0 + \epsilon.$$

Linear Regression

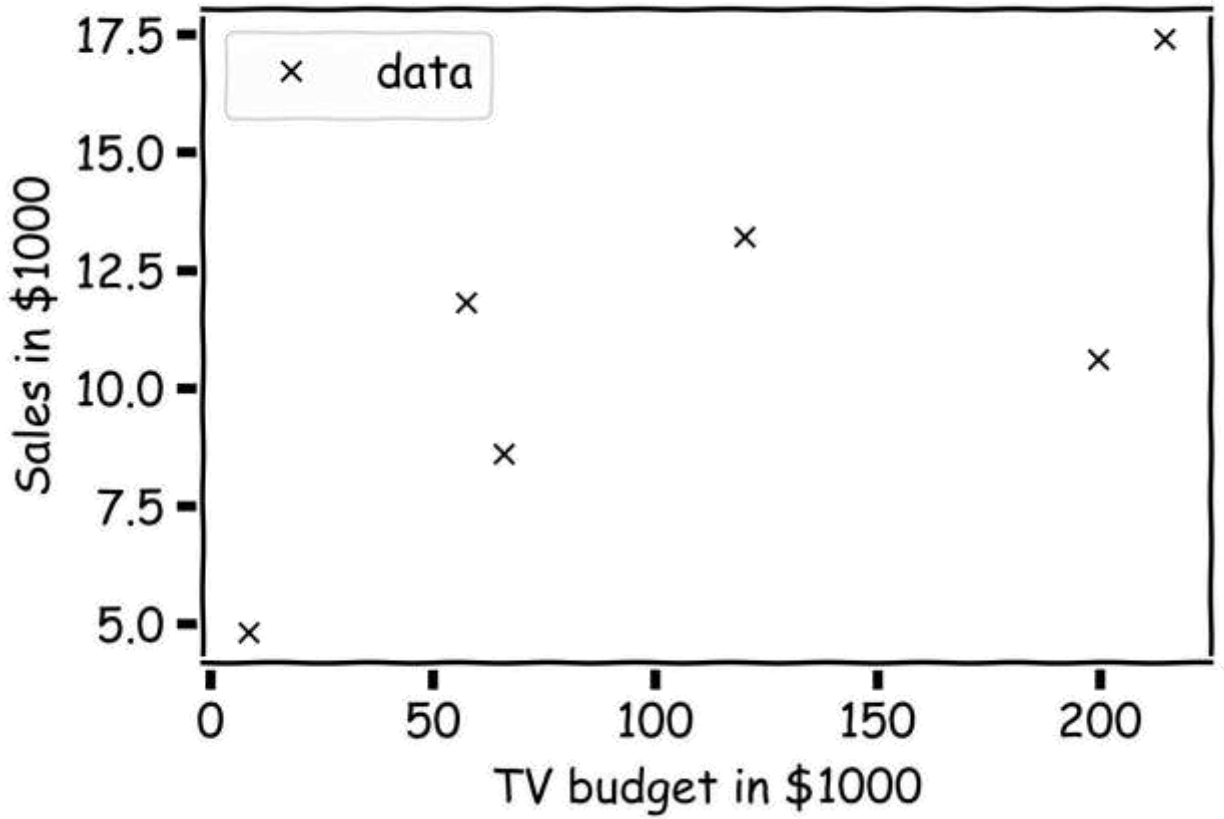
- … then it follows that our estimate is:

$$\widehat{Y} = \widehat{f}(X) = \widehat{\beta_1} X + \widehat{\beta_0}$$

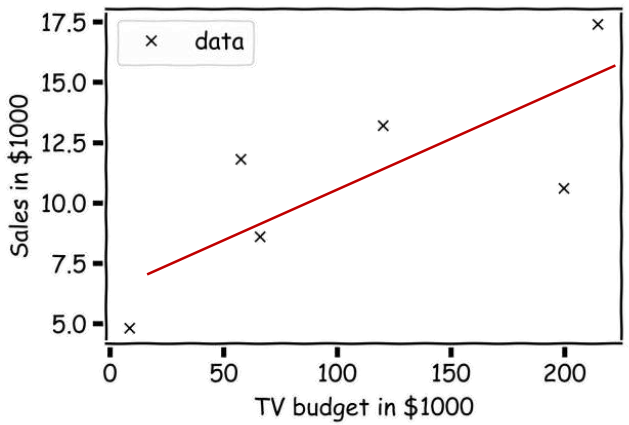
- where $\hat{\beta}_1$ and $\hat{\beta}_0$ are **estimates** of β_1 and β_0 respectively, that we compute using observations.

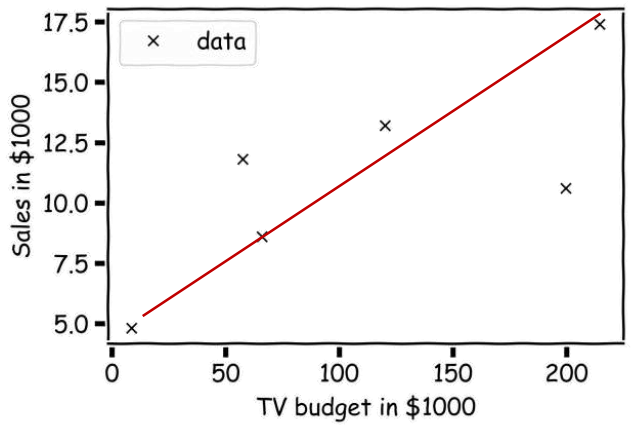
Estimate of the regression coefficients

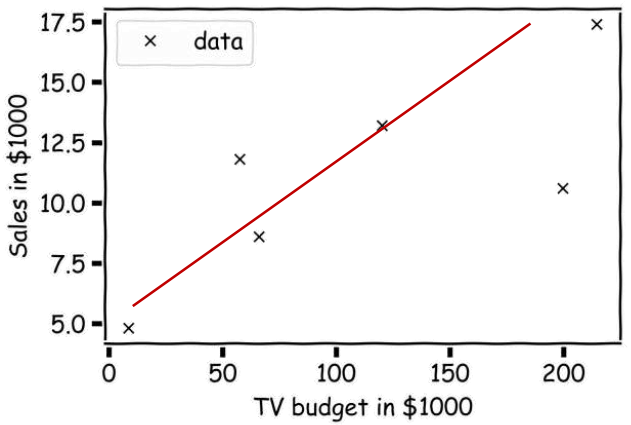
For a given data set



Estimate of the regression coefficients (cont)





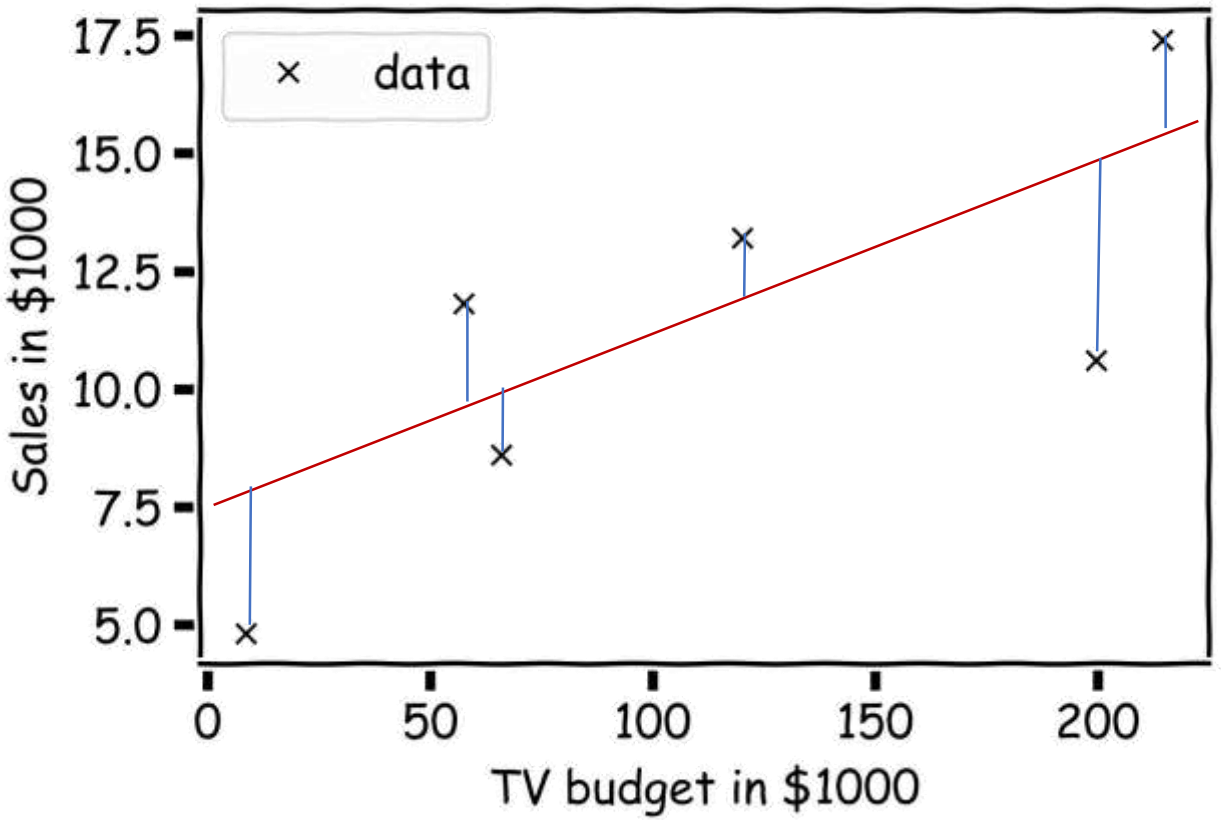


Which of the above three lines fit the data points the best?

a. One which goes through maximum number of points
b. One with least slope
c. One from which no point is too far, i.e. it is approximately in middle of all points

Estimate of the regression coefficients (cont)

To compute the best fit, we first calculate the residuals



Python package

```
X=market_data[['TV','radio','newspaper']]
Y=market_data['sales']
model= sklearn.linear_model.LinearRegression()
model.fit(X,Y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
X_new=[[147,23,30.55]]
print(model.predict(X_new))
```

```
[13.97078723]
```

So how do Linear Regression solvers work?

- Matrix Methods
  - Exact methods that solve the set of linear equations
  - Involve computation of matrix inverse or pseudoinverse (more efficient)
- Gradient Descent
  - A generic method of solving optimization problems
  - Begin with a random point and reach the optimal solution through a sequence of improvements
  - Faster improvements could be done by stochastic methods

Matrix Algebra for n-dimensions

- Loss (L) = MSE(β) = $\sum_{i=1}^{N}(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2$
- MSE(β) = $(\boldsymbol{y} - \boldsymbol{X\beta})^T(\boldsymbol{y} - \boldsymbol{X\beta})$, where **X** is a nx(p+1) matrix with each row as an input vector (including '1' for the intercept) and **y** is a n dimensional vector of the outputs in the training set
- To minimize, we differentiate with respect to β and we get
  - $(\boldsymbol{X})^T(\boldsymbol{y} - \boldsymbol{X\beta})$=0
- If $\boldsymbol{X^T X}$ is non-singular, meaning, inverse exists, then
  - **β** = $(\boldsymbol{X^T X})^{-1}(\boldsymbol{X^T y})$

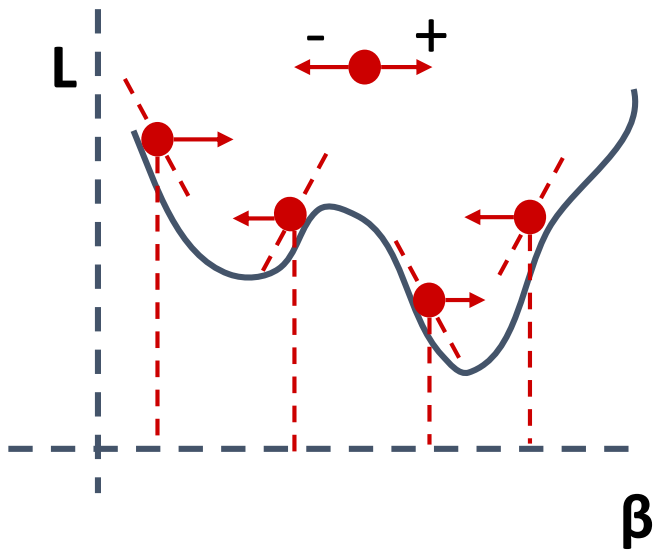
Matrix Algebra for n-dimensions

- Computational complexity of computing the matrix inverse: $O(p^{2.4})$ to $O(p^3)$ depending on implementation.
- Scikit-learn's Linear Regression class uses SVD approach $O(p^2)$
  - SVD stands for singular value decomposition
  - Uses pseudoinverse approach (Moore-Penrose) :numpy.linalg.pinv()
  - $\boldsymbol{\beta} = (\boldsymbol{X}^{+}\boldsymbol{y})$
- They both have linear complexity in terms of the number of instances, n; but at least quadratic in p
- So, we need to look at alternate techniques if p is very large, e.g., 100,000

Gradient Descent Approach

- Start from a random point, i.e. generate a random **β**
  1. Determine which direction will reduce the MSE
  2. Compute the slope of the function (its derivative) at this point and go in the reverse direction
  3. λ is the learning rate parameter
  4. Go to #1, until convergence, i.e. MSE is minimized
- For Linear regression, MSE is a convex function
  - There is no local minima, just a global minimum
  - Continuous and slope that never changes abruptly

$$\boldsymbol{\beta}^{(i+1)} = \boldsymbol{\beta}^{(i)} - \lambda \frac{d\mathrm{L}}{d\boldsymbol{\beta}}$$



Stochastic Approaches

- Batch GD update equation $\boldsymbol{\beta}^{(i+1)} = \boldsymbol{\beta}^{(i)} - \lambda \frac{2}{\mathrm{p}} (\boldsymbol{X})^T (\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{Y})$
  - Uses the whole batch of training data at each gradient step
- Stochastic GD
  - Picks only a random instance of training data to update gradients
  - Causes irregular descent, but better chance of finding global minimum
  - Simulated annealing: Reduce the learning rate gradually to reduce irregularity
- Mini-batch GD
  - Small set of random instances of training data are used

Parametric or Non-Parametric?

	Linear Regression (parametric)	k-NN Approach (non-parametric)
Assumption on function f	A linear function is assumed	Can work even if the function is non-linear. But it has to be locally constant
High dimensions	Complexity problems which can be overcome by efficient algorithms	Difficult to find neighbors nearby which can cause errors
Bias	Low (if true function is linear) High (if true function is non-linear)	Small K=> Low bias Large K => High Bias
Variance	Depends on the problem	Small K=> High Variance Large K => Low Variance
Computations	Once during the model fitting phase. After that predictions are quick	Every time a prediction has to be made, we look at all the training points

kNN: Kernel Regression

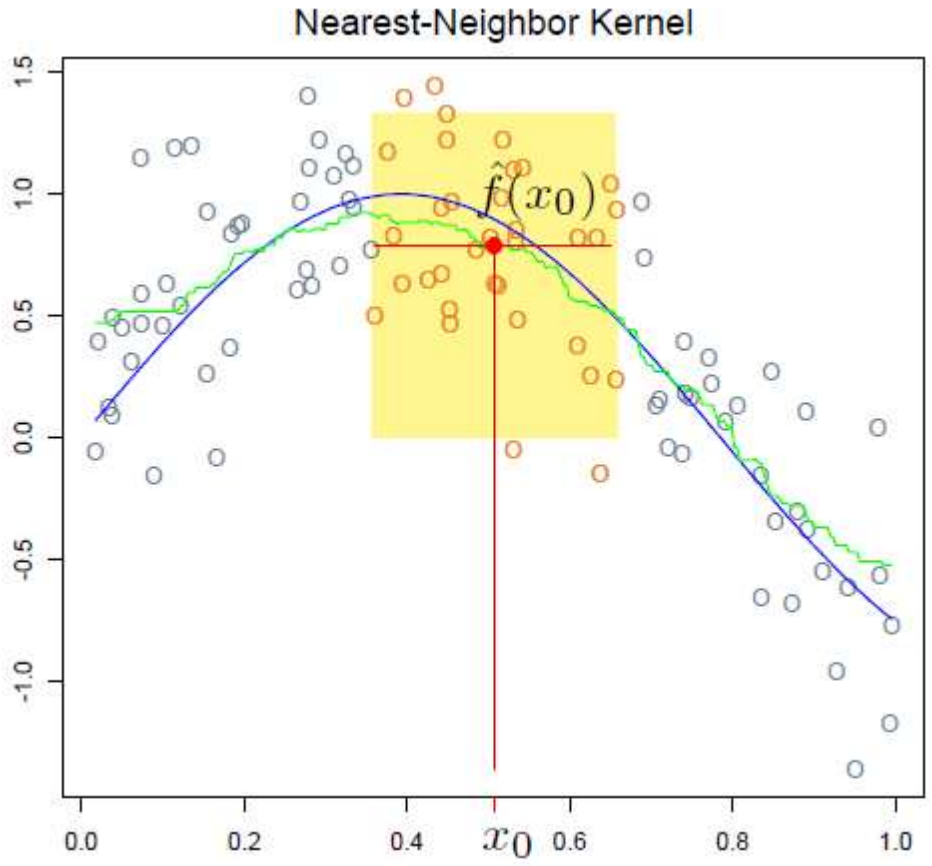
- What if we took all the points, not just the k nearest points, and introduced a weighting function that weights by distance (so that we weight the value of closer points more)?

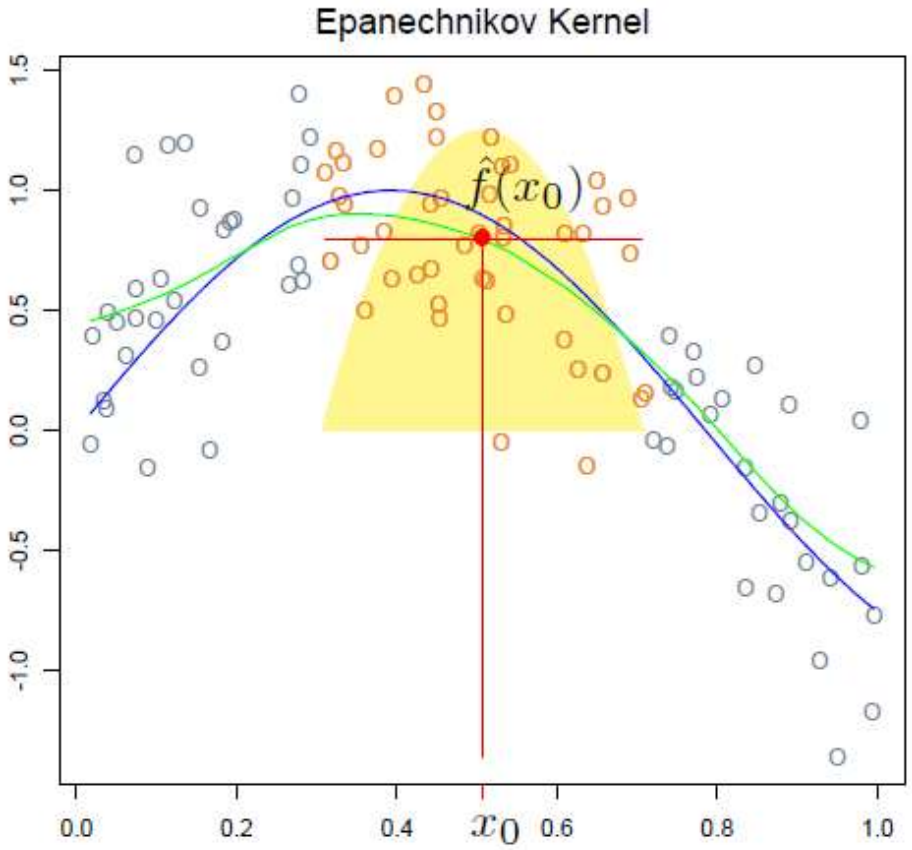
$$\hat{y}^* = \frac{\sum_n K(x^*, x_n) y_n}{\sum_n K(x^*, x_n)}$$

- Traditional kNN method can be seen as a 0/1 weighting model
- Examples of kernels: Epanechnikov quadratic kernel, Tri-cube Kernel, and Gaussian Kernel.

Comparison of Kernels

$$K(x) = \frac{3}{4}(1 - x^2)\ for\ -1 \le x \le 1$$





Confidence intervals for predictor estimators

- What causes errors in estimation of $\boldsymbol{\beta_i}$?
  - $\boldsymbol{\varepsilon}$ $(\text{noise in the model})$
  - we do not know the exact form of $f(x)$
  - limited sample size
- Variance of $\widehat{\beta}_i$ is called as ***standard error***, $SE(\hat{\beta}_i)$
- To estimate SE, we use **Bootstrapping**
  - sampling from the training data *(X,Y) to* estimate its statistical properties.
- In our case, we can sample with replacement
  - Compute $\hat{\beta}_i$multiple times by random sampling
  - Variance of multiple estimates approximates the true variance

Standard Errors Intuition from Formulae

- **Better model:** $(\hat{f} - y_i) \downarrow \Longrightarrow \sigma \downarrow \Longrightarrow SE \downarrow$

$$\sigma \approx \sqrt{\sum \frac{(\hat{f}(x) - y_i)^2}{n-2}}$$

- **More data:** $n \uparrow$ and $\sum_i (x_i - \bar{x})^2 \uparrow \Longrightarrow SE \downarrow$
- **Larger coverage**: $var(x)$ or $\sum_i (x_i - \bar{x})^2 \uparrow \Longrightarrow SE \downarrow$
- **Better data:** $\sigma^2 \downarrow \Rightarrow SE \downarrow$

$$SE\left(\widehat{\beta}_0\right) = \sigma\sqrt{\frac{1}{n} + \frac{\overline{x}^2}{\sum_i (x_i - \overline{x})^2}}$$

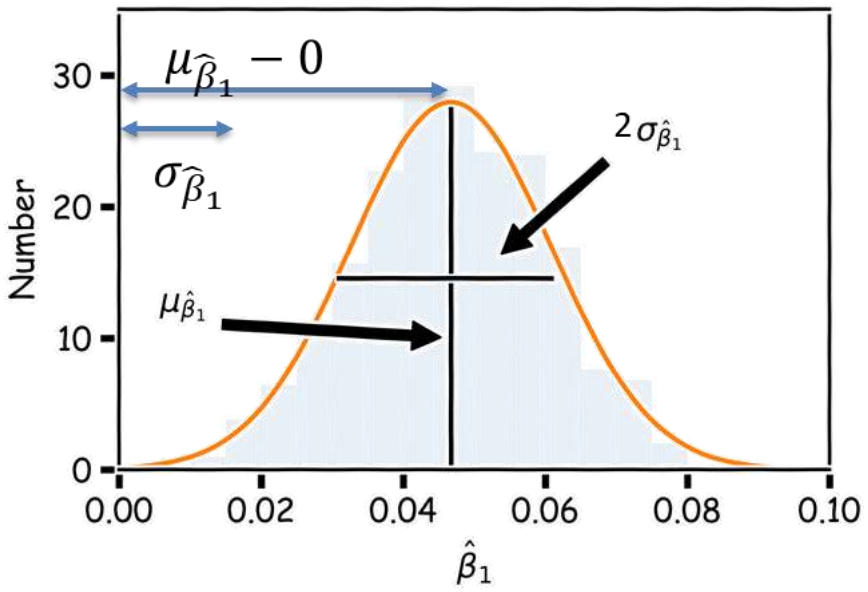
$$SE\left(\widehat{\beta}_1\right) = \frac{\sigma}{\sqrt{\sum_i (x_i - \overline{x})^2}}$$

$$General\ Formula\colon\ SE(\beta)^2 = \sigma^2 (X^T X)^{-1}$$

Significance of predictor variables

- As we saw, there are inherent uncertainties in estimation of **β**
- We evaluate the importance of predictors using hypothesis testing, using the t-statistics and p-values (Small p-value(<0.05) => significant)
- Null hypothesis is that $\beta_i=0$

Test statistic here would be $t = \frac{\mu_{\hat{\beta}_1}}{\sigma_{\hat{\beta}_1}}$
Which measures the distance of the mean from zero in units of standard deviation.



Sample Results

```
import statsmodels.api as sm
est = sm.OLS(y, X2)
est2 = est.fit()
print(est2.summary())
```

OLS Regression Results

Dep. Variable:	sales	R-squared:	0.897
Model:	OLS	Adj. R-squared:	0.896
Method:	Least Squares	F-statistic:	570.3
Date:	Wed, 28 Nov 2018	Prob (F-statistic):	1.58e-96
Time:	22:06:58	Log-Likelihood:	-386.18
No. Observations:	200	AIC:	780.4
Df Residuals:	196	BIC:	793.6
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.9389	0.312	9.422	0.000	2.324	3.554
x1	0.0458	0.001	32.809	0.000	0.043	0.049
x2	0.1885	0.009	21.893	0.000	0.172	0.206
x3	-0.0010	0.006	-0.177	0.860	-0.013	0.011

Omnibus:	60.414	Durbin-Watson:	2.084
Prob(Omnibus):	0.000	Jarque-Bera (JB):	151.241
Skew:	-1.327	Prob(JB):	1.44e-33
Kurtosis:	6.332	Cond. No.	454.

R^2, p-value and F-statistic

Subset Selection Techniques

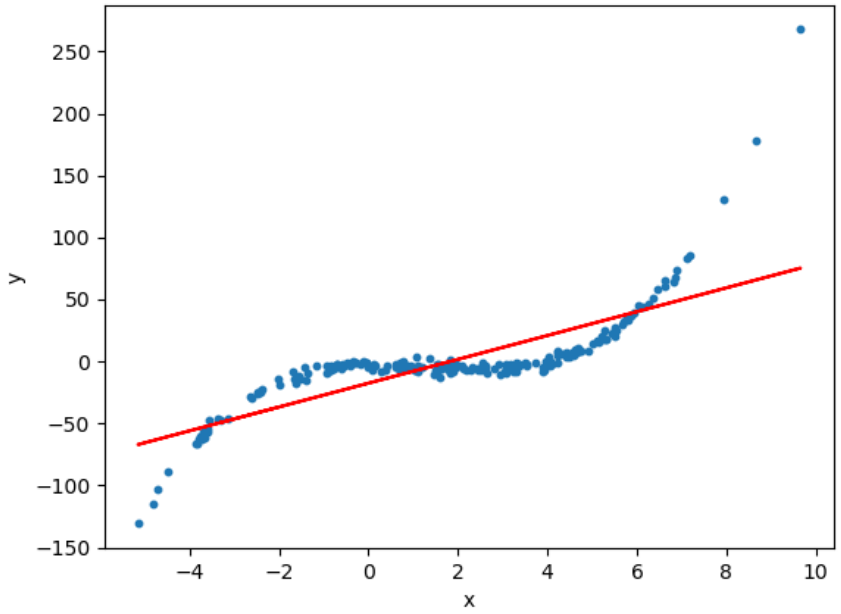
- **Forward selection:**
  - Begin will a null set, S
  - Perform J linear regressions, each with exactly one variable
  - Add the variable that results in lowest Cross-validation error to the set, S
  - Again, perform J-1 linear regressions with 2 variables
  - Add the variable that results in lowest Cross-validation error to the set, S
  - Continue until some stopping criteria is reached… eg. CV error is not decreasing
- **Backward selection** begins with all the variables and removes the variable with highest p-value at successive steps
- **Mixed selection** is similar to Forward Selection, but it may also remove a variable if it doesn't yield any improvement to the model

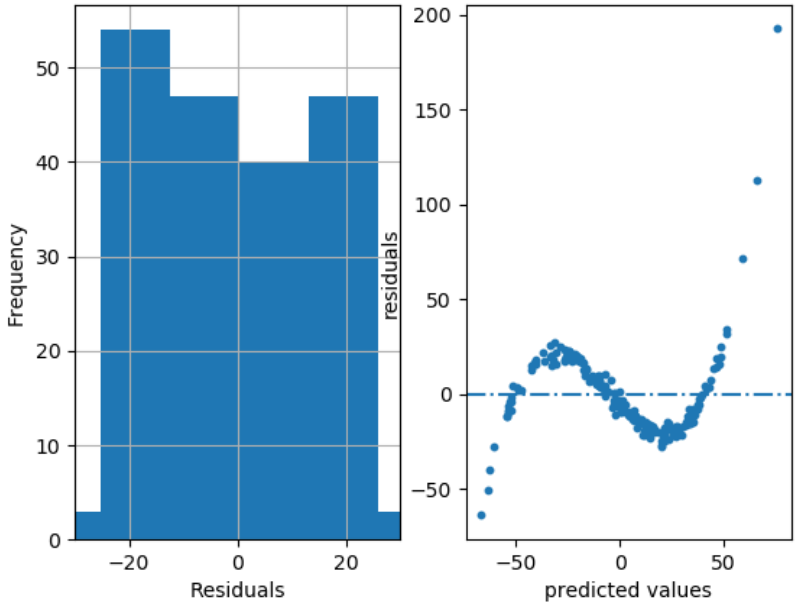
Do I need more predictors/change of model?

- When we estimated the variance of ϵ, we assumed that the residuals $r_i = y_i - \hat{y}_i$ were uncorrelated and normally distributed with mean 0 and fixed variance.
- These assumptions need to be verified using the data. In residual analysis, we typically create two types of plots:

1. a plot of r_i with respect to x_i or $\hat{y}_i$. This allows us to compare the distribution of the noise at different values of x_i.
2. a histogram of r_i. This allows us to explore the distribution of the noise independent of x_i or $\hat{y}_i$.

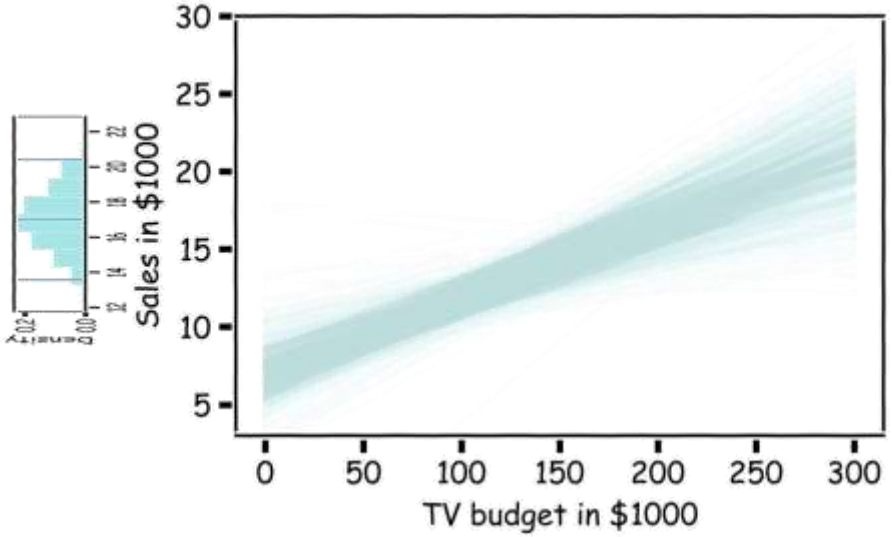
Patterns in Residuals

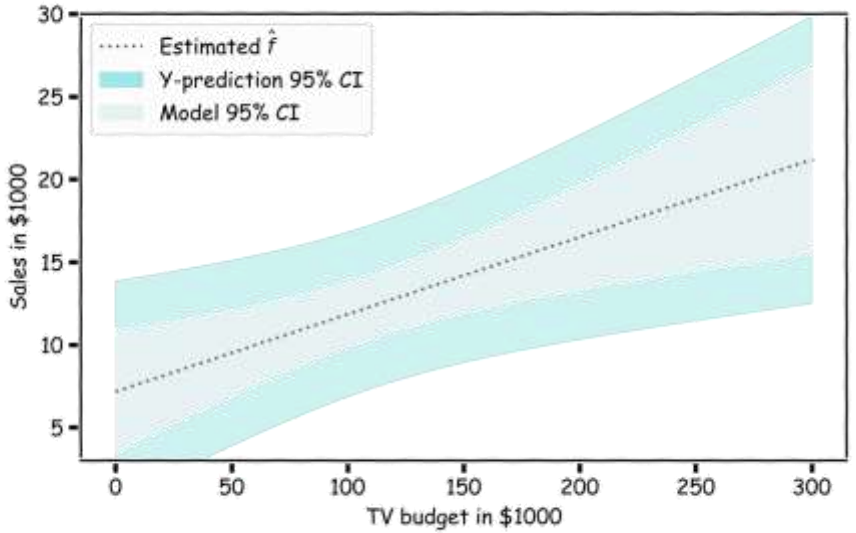




- Residuals are easier to interpret than the model
- We plot $(y_i - \hat{y}_i)$ with y_i, so the graph is always 2-D

Confidence intervals on predictions of y





- Depends on confidence on **β**
- Different **β** => different values of y
- Given x, examine distribution of $\hat{f}$, determine the mean and standard deviation.
- For each of these $f(x)$, the prediction for $y \sim N(f, \sigma_\epsilon)$

Potential problems of Linear Models

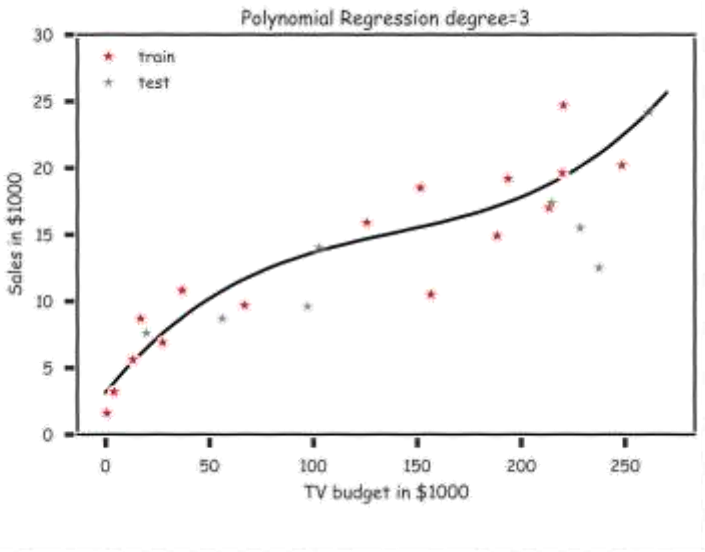
- Non-linearity
  - Can use polynomial linear regression or design better features
- Outliers
  - Disturbs the models because of quadratic penalty, Discard outliers carefully
- High-leverage points
  - Outliers in the predictor variables
- Collinearity (2 or more predictor variables have high correlation)
  - Keep one of them or design a good combined feature
- Correlation of error terms, Non-constant variance of error terms
  - Gives higher confidence in the model, can’t trust the CI on model parameter

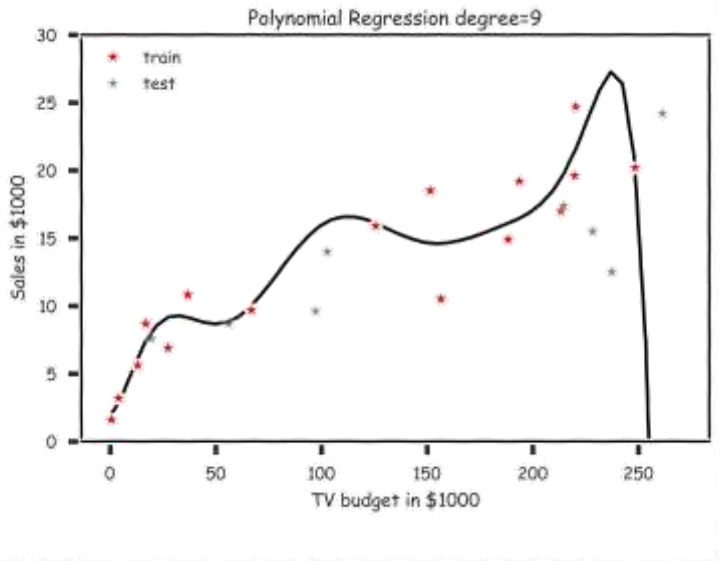
Polynomial Regression

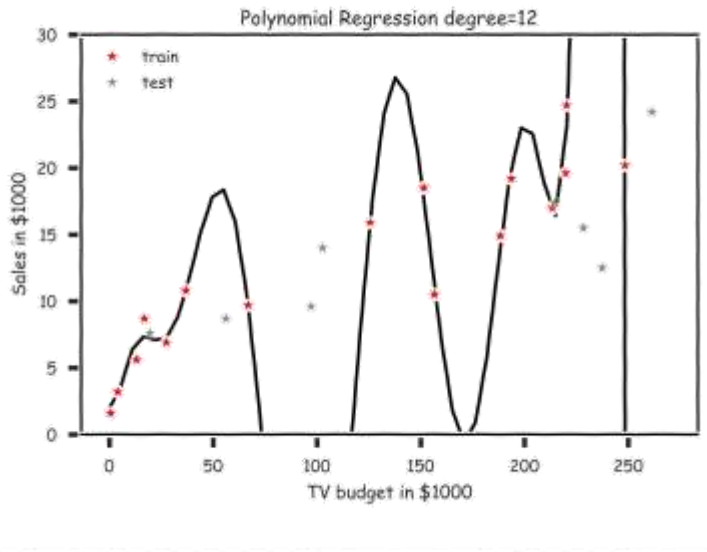
- The simplest non-linear model we can consider, for a response *Y* and a predictor *X*, is a polynomial model of degree *M*, $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_M x^M + \epsilon.$
- Just as in the case of linear regression with cross terms, polynomial regression is a special case of linear regression - we treat each x^m as a separate predictor. Thus, we can write

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} 1 & x_1^1 & \ldots & x_1^M \\ 1 & x_2^1 & \ldots & x_2^M \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \ldots & x_n^M \end{pmatrix}, \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{pmatrix}.$$

Polynomial Regression

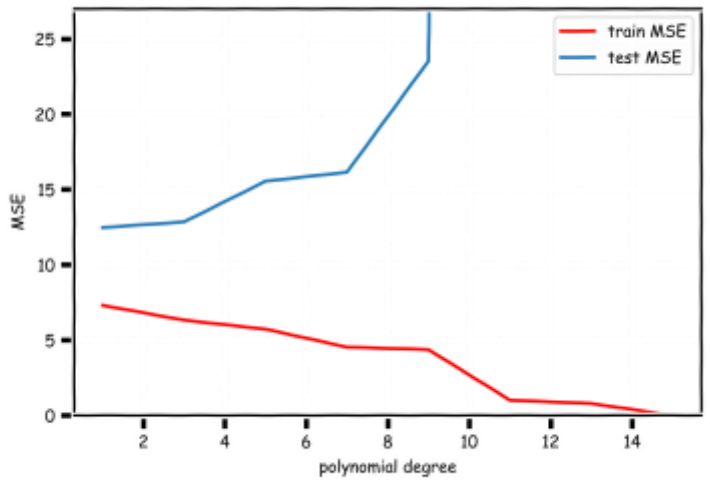






- Which of the above three is the best model?
  - Check RMSE
  - Check R^2
  - Remember bias and variance??

Benefit of Cross-Validation



$$CV(\text{Model}) = \frac{1}{K}\sum_{i=1}^{K} L(\hat{f}_{C_{-i}}(C_i))$$

- Using cross-validation, we generate validate the models on a portion of training data which our learning algorithm has never seen.
- Leave-one out method is used when the number of sample points is very small.

Regularization of Linear Models

- Goal: Reduce over-fitting of the data by reducing degrees of freedom
- For a linear model, regularization is typically achieved by constraining the weights of the model

$$L_{reg}(\beta) = L(\beta) + \lambda R(\beta),$$

where λ is a scalar that gives the weight (or importance) of the regularization term.

- Fitting the model using the modified loss function L_{reg} would result in model parameters with desirable properties (specified by R).

Ridge Regression

- Alternatively, we can choose a regularization term that penalizes the squares of the parameter magnitudes. Then, our regularized loss function is:

$$L_{Ridge}(\beta) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda \sum_{j=1}^{J} \beta_j^2.$$

- Works best when least-square estimates have high variance
- As λ increases, flexibility decreases, variance decreases, bias increases slightly

- Note that $\sum_{j=1}^{J} |\beta_j|^2$ is the $\boldsymbol{l_2}$ norm of the vector $\boldsymbol{\beta}$ $\sum_{j=1}^{J} \beta_j^2 = \|\boldsymbol{\beta}\|_2^2$

Ridge Regression

- We often say that L_{ridge} is the loss function for $\boldsymbol{l_2}$ regularization.
- Finding the model parameters $\boldsymbol{\beta}_{\text{ridge}}$ that minimize the $\boldsymbol{l_2}$ regularized loss function is called ***ridge regression.***

In [ ]:
```
from sklearn.linear_model import Ridge
```

In [20]:
```
X_train = train[all_predictors].values
X_val = validation[all_predictors].values
X_test = test[all_predictors].values

ridge_regression = Ridge(alpha=1.0, fit_intercept=True)
ridge_regression.fit(np.vstack((X_train, X_val)), np.hstack((y_train, y_val)))

print('Ridge regression model:\n {} + {}^T . x'.format(ridge_regression.intercept_, ridge_regression.coe
```

```
Ridge regression model:
 -525.7662550875951 + [ 0.24007312  8.42566029  2.04098593 -0.04449172 -0.01227935  0.41902475
 -0.50397312 -4.47065168  4.99834262  0.          0.          0.29892679]^T . x
```

In [21]:
```
print('Train R^2: {}, test R^2: {}'.format(ridge_regression.score(np.vstack((X_train, X_val)),
                                                                  np.hstack((y_train, y_val))),
                                           ridge_regression.score(X_test, y_test)))
```

```
Train R^2: 0.5319764744847737, test R^2: 0.7881798111697319
```

LASSO (least absolute shrinkage and selection operator) Regression

- Ridge regression reduces the parameter values but doesn't force them to go to zero. LASSO is very effective in doing that.
- It uses the following regularized loss function is:

$$L_{LASSO}(\beta) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \boldsymbol{\beta}^\top \boldsymbol{x}_i|^2 + \lambda\sum_{j=1}^{J}|\beta_j|.$$

- Note that $\sum_{j=1}^{J}|\beta_j|$ is the $\boldsymbol{l_1}$ norm of the vector $\boldsymbol{\beta}$

$$\sum_{j=1}^{J}|\beta_j| = \|\boldsymbol{\beta}\|_1$$

LASSO Regression

- Hence, we often say that L_{LASSO} is the loss function for l_1 regularization.
- Finding the model parameters $\boldsymbol{\beta}_{LASSO}$ that minimize the l_1 regularized loss

In [ ]:
```
from sklearn.linear_model import Lasso
```

In [22]:
```
lasso_regression = Lasso(alpha=1.0, fit_intercept=True)
lasso_regression.fit(np.vstack((X_train, X_val)), np.hstack((y_train, y_val)))

print('Lasso regression model:\n {} + {}^T . x'.format(lasso_regression.intercept_, lasso_regression.coe
```

```
Lasso regression model:
 10.424895873901445 + [ 0.24482603  3.48164594  1.84836859 -0.06864603 -0.         -0.
 -0.02249766 -0.          0.          0.          0.          ]^T . x
```

In [23]:
```
print('Train R^2: {}, test R^2: {}'.format(lasso_regression.score(np.vstack((X_train, X_val)),
                                                                  np.hstack((y_train, y_val))),
                                           lasso_regression.score(X_test, y_test)))
```

```
Train R^2: 0.48154992527975765, test R^2: 0.6846451270316087
```

Choosing λ

- In both ridge and LASSO regression, we see that the larger our choice of the **regularization parameter** λ, the more heavily we penalize large values in **β,**
- If λ is close to zero, we recover the MSE, i.e. ridge and LASSO regression is just ordinary regression.
- If λ is sufficiently large, the MSE term in the regularized loss function will be insignificant and the regularization term will force $\boldsymbol{\beta}_{\text{ridge}}$ and $\boldsymbol{\beta}_{\text{LASSO}}$ to be close to zero.
- To avoid ad-hoc choices, we should select λ using cross-validation.
- Once the model is trained, we use the unregularized performance measure to evaluate the model's performance.

Elastic Net

- Middle ground between Ridge and Lasso regression
- Regularization term is a simple mix with parameter ‘r’

- Elastic Net has better convergence features over Lasso.

```
Sklearn.linear_model import ElasticNet
Elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)
```