Section 4. Statistics of Machine Learning

I. Covariance of Parameters

II. Confidence Intervals for Coefficients

III. Subset Selection Methods

IV. Gauss-Markov Theorem

Case Study: House Price Example:

BEDS	BATHS	LOCATION	SQUARE_FEET	LOT_SIZE	YEAR_BUILT	PRICE
3	3	Newton	2969	15014	1967	1090000
3	2.5	Newton	1566	5582	1922	805000
4	2.5	Newton Corner	2532	6273	1953	905000
7	4.5	Newton Center	6748	26607	1902	2660000
...	....	...	...	...	...	...

The **features** of the data are

BEDS – x_1 Number of bedrooms
BATHS - x_2 Number of bathrooms
LOCATION – x_3 city/town
SQUARE_FEET – x_4 Square feet of the living spaces
LOT_SIZE - x_5 Square feet of the lot size.
YEAR_BUILT - x_6

Training Data: $D = \{(\vec{x}^{(1)}, y^{(1)}), \ldots, (\vec{x}^{(n)}, y^{(n)})\}$

Assumption: Linear Model $h(\vec{x}) = \vec{\theta}^T \vec{x} = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$

feature x_i

Which parameters θ_i correspond to statistically significant features?

Which parameters θ_i correspond to less significant features? i.e. $\theta_i \approx 0$.

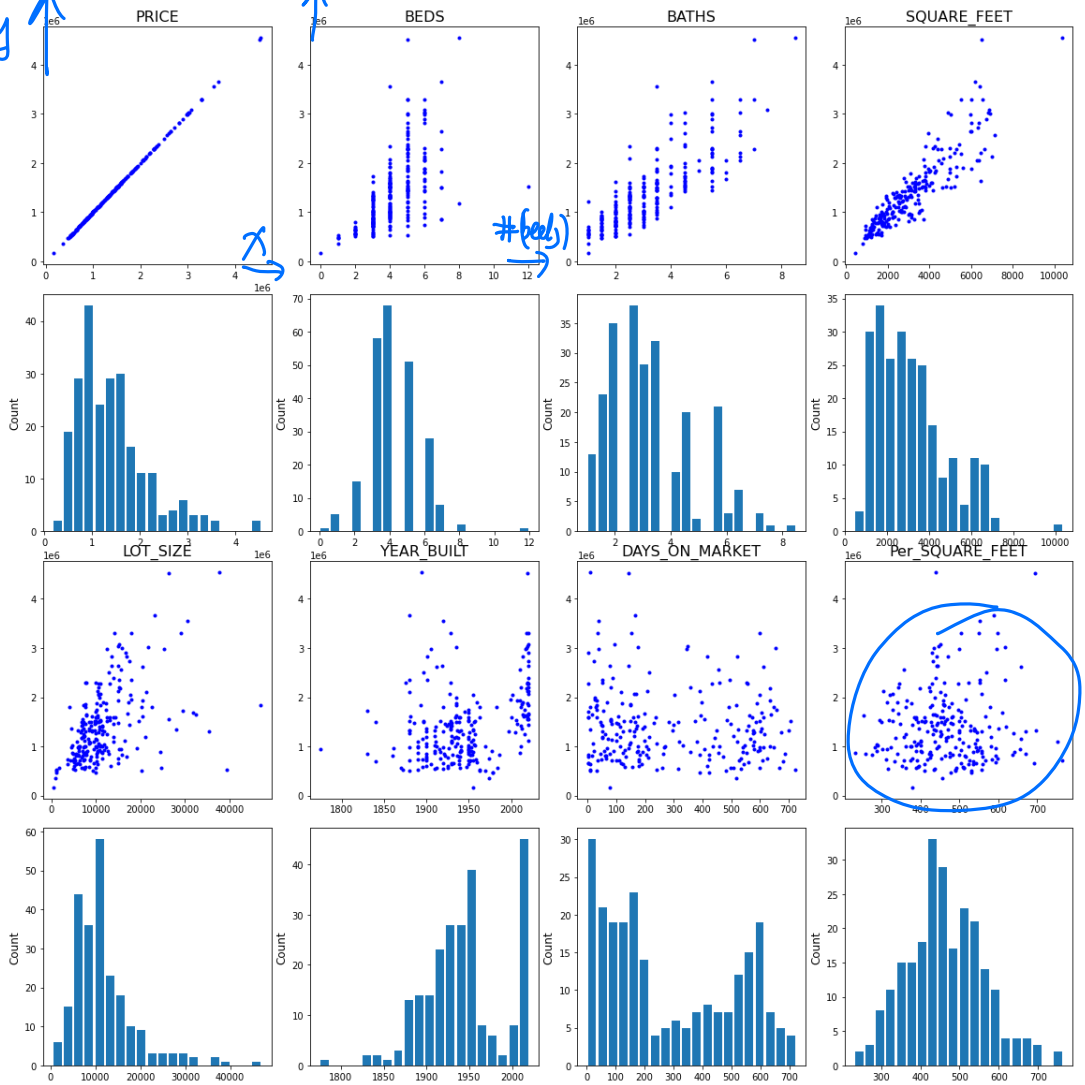
Data X y

• "pandas"

DataFrame.describe()

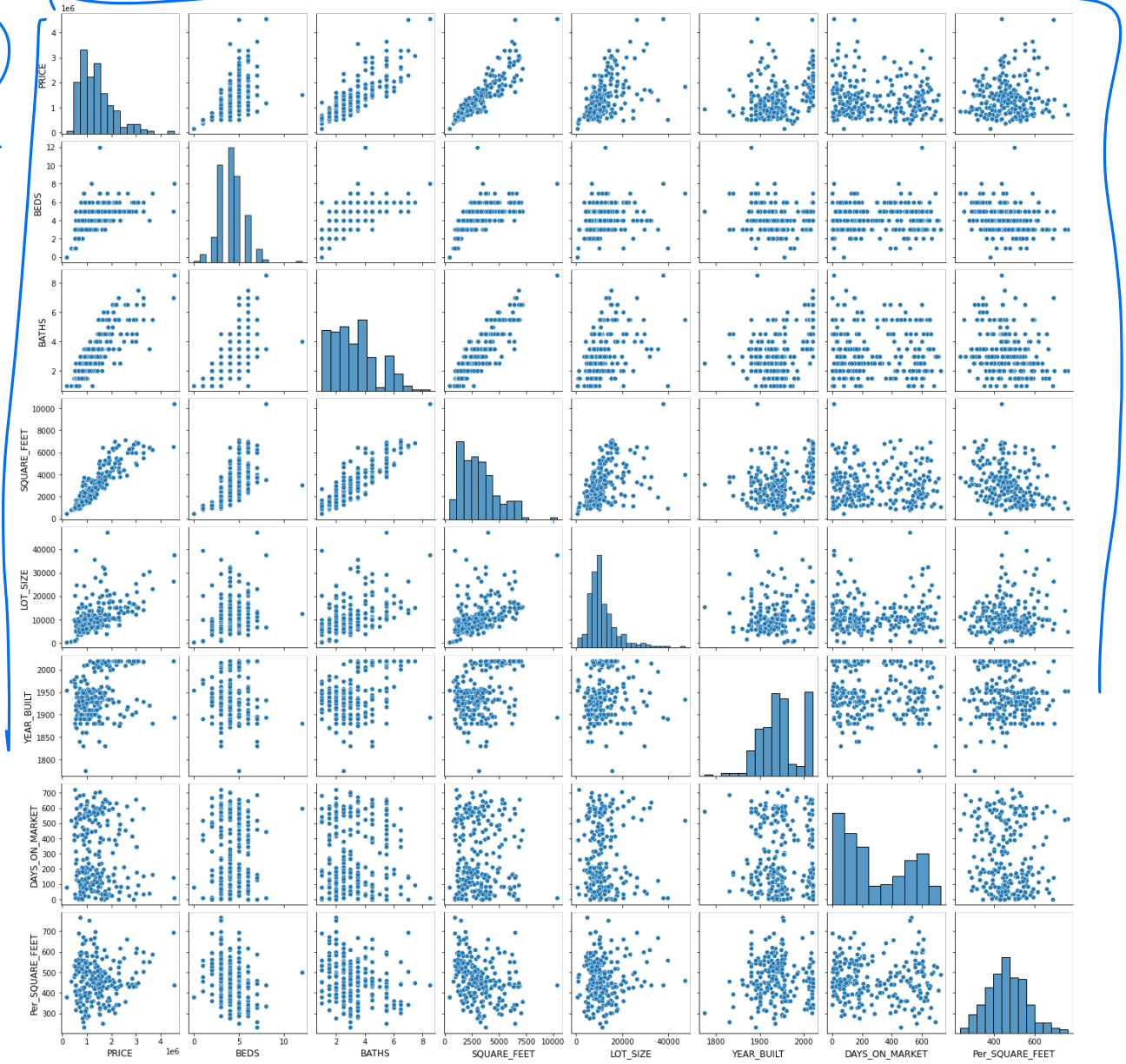
	BEDS	BATHS	SQUARE_FEET	LOT_SIZE	YEAR_BUILT	DAYS_ON_MARKET	PRICE
count	58	58	58	58	58	58	58
mean	4.293103	3.284483	3330.379	12709.79	1929.552	14.53448	1595263
std	1.297888	1.47824	1573.35	7656.194	46.82786	8.364955	807219
min	2	1.5	840	3966	1830	1	630000
25%	3	2.5	1906.75	7365.75	1897.75	6.25	941250
50%	4	3	3143.5	10146.5	1926.5	15	1324250
75%	5	4	4363.5	15004.5	1953	21	1893750
max	8	8.5	6748	33843	2020	29	3550000

Matplotlib


PRICE
BEDS
BATHS
SQUARE_FEET
LOT_SIZE
YEAR_BUILT
DAYS_ON_MARKET
Per_SQUARE_FEET
Count
1e6

seaborn

easy



I. Covariance/Variance matrix

For a d- dimensional vector $\vec{x}$ of random variables, the **covariance** of $\vec{x}$ is a $d \times d$ symmetric matrix:

$$\begin{aligned} \mathrm{Cov}(\vec{x}) &= E[(\vec{x} - E[\vec{x}])(\vec{x} - E[\vec{x}])^T] \\ &= E[\vec{x}\vec{x}^T - E[\vec{x}]E[\vec{x}]^T] \end{aligned}$$

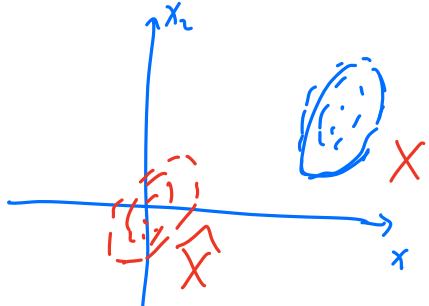
The (i,j)-entry of $\mathrm{Cov}(\vec{x})$ given by the **covariance** of x_i and x_j

$$\mathrm{Cov}(\vec{x})_{ij} = \mathrm{Cov}(x_i, x_j) = E[(x_i - E[x_i])(x_j - E[x_j])]$$
$$= E[x_i x_j - E[x_i]E[x_j]] = E(x_i x_j) - E(x_i)E(x_j)$$

In particular, the diagonal entries of $\mathrm{Cov}(\vec{x})$ given by the **variance** of x_i.

$$\mathrm{Cov}(\vec{x})_{ii} = \mathrm{Var}(x_i) = \sigma(x_i)^2$$

Data $X = \begin{bmatrix} - & \vec{x}^{(1)T} & - \\ & \vdots & \\ - & \vec{x}^{(n)T} & - \end{bmatrix}$

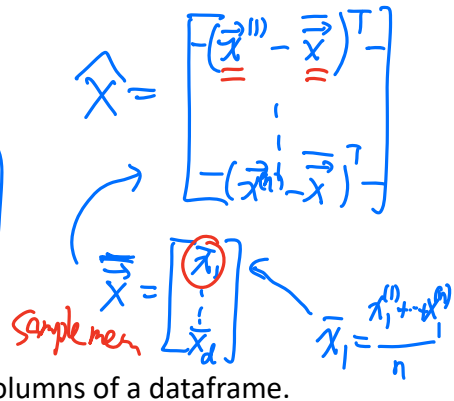


For a data set $D = \{(\vec{x}^{(1)}, y^{(1)}), \ldots, (\vec{x}^{(n)}, y^{(n)})\}$ from random variables $\vec{x}$

The **sample covariance** is calculated (estimate) by

$$\mathrm{Cov}(x_i, x_j) = \frac{1}{n-1}\sum_{k=1}^{n}(x_i^{(k)} - \bar{x}_i)(x_j^{(k)} - \bar{x}_j)$$

$$\Leftrightarrow \quad Cov(\vec{X}) = \frac{1}{n-1}\hat{X}^T\hat{X}$$



In pandas lib, DataFrame.cov() is a function that compute the covariance between the columns of a dataframe.

The **covariance** in the variables for the dataset:

	BEDS	BATHS	SQUARE_FEET	LOT_SIZE	YEAR_BUILT	DAYS_ON_MARKET	PRICE
BEDS	1.684513	1.248488	1.521168e+03	5.018816e+03	1.923170	-3.843618	7.160882e+05
BATHS	1.248488	2.185194	1.925302e+03	4.943595e+03	28.884150	-4.689806	8.206805e+05
SQUARE_FEET	1521.167	1925.3024	2.475431e+06	7.843136e+06	19298.330913	-5634.118572	1.139606e+09
LOT_SIZE	5018.816	4943.5949	7.843136e+06	5.861731e+07	-57213.2171	-12123.115547	4.430783e+09
YEAR_BUILT	1.923170	28.884150	1.929833e+04	-5.7213e+04	2192.848155	-84.615850	5.700393e+06
DAYS_ON_MARKET	-3.843618	-4.689806	-5.6341e+03	-1.21231e+04	-84.615850	69.972474	-2.3119e+06
PRICE	716088.24	820680.462	1.1396e+09	4.4307e+09	5.7003e+06	-2.3119e+06	6.51602e+11

➢ Correlation matrix.

x_i, x_j independent $\Rightarrow$ Corr.(x_i, x_j)=0

The **correlation** of $\vec{x}$ is a $d \times d$ matrix $\mathrm{Corr}(\vec{x})$ defined as

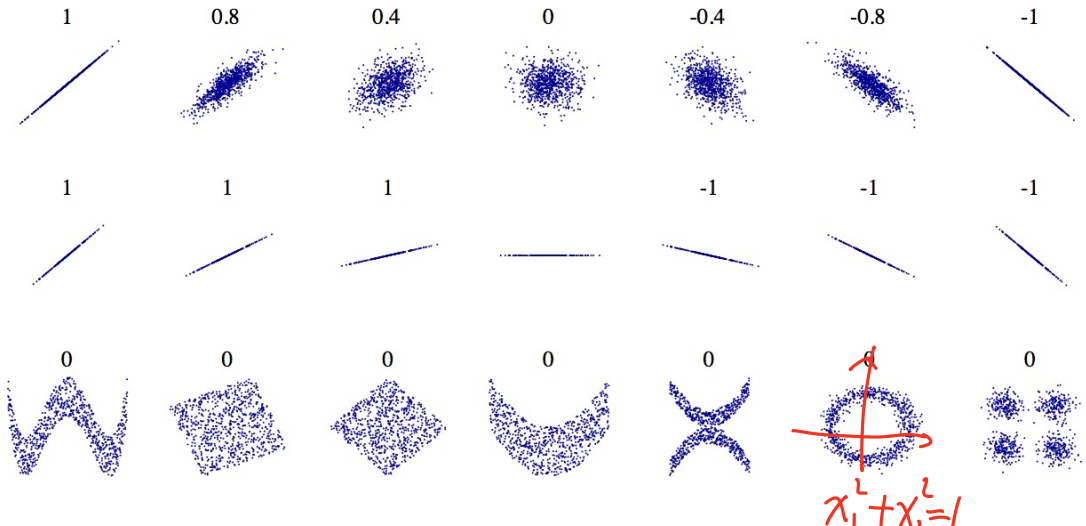
$$\mathrm{Corr}(\vec{x})_{ij} = \frac{\mathrm{Cov}(\vec{x})_{ij}}{\sigma(x_i)\sigma(x_j)}$$

Remark:

1. Correlation matrix is the covariance matrix of the standardized random variables $\frac{x_i}{\sigma(x_i)}$

2. The correlation coefficient ranges from –1 to 1. When it is close to 1, it means that there is a strong positive correlation. Coefficients close to zero mean that there is no linear correlation.

3. The correlation coefficient only measures linear correlations. It may completely miss out on nonlinear relationships.

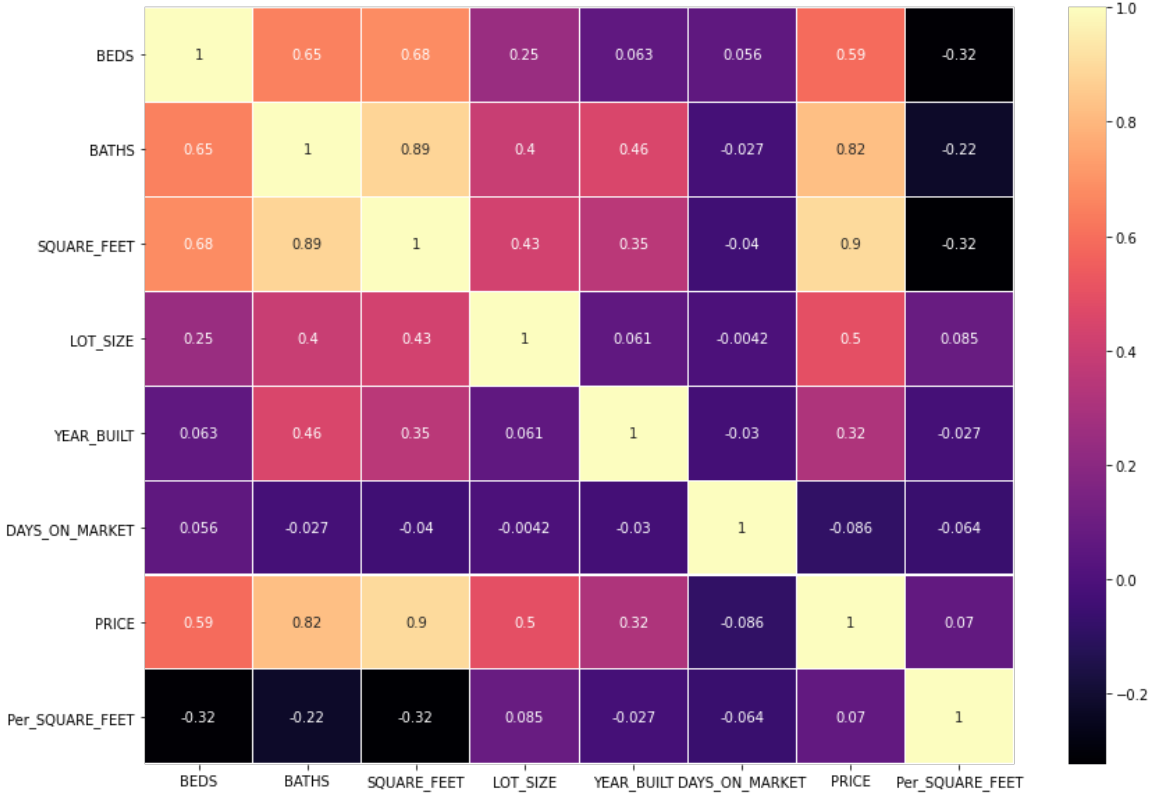


The **correlation** of the variables for the dataset: (by function DataFrame.corr())

	BEDS	BATHS	SQUARE_FEET	LOT_SIZE	YEAR_BUILT	DAYS_ON_MARKET	PRICE
BEDS	1.000000	0.650732	0.744928	0.505070	0.031643	-0.354029	0.683499
BATHS	0.650732	1.000000	0.827806	0.436802	0.417263	-0.379268	0.687761
SQUARE_FEET	0.744928	0.827806	1.000000	0.651106	0.261933	-0.428092	0.897301
LOT_SIZE	0.505070	0.436802	0.651106	1.000000	-0.159580	-0.189294	0.716929
YEAR_BUILT	0.031643	0.417263	0.261933	-0.159580	1.000000	-0.216015	0.150803
DAYS_ON_MARKET	-0.354029	-0.379268	-0.428092	-0.189294	-0.216015	1.000000	0.342381
PRICE	0.683499	0.687761	0.897301	0.716929	0.150803	-0.342381	1.000000

```
import pandas as pd
import seaborn as sns
fig,ax = plt.subplots(figsize=(14, 10))

#sns.set(font_scale=1.4)
sns.heatmap(Newton1.corr(), ax=ax, linewidths=0.05,cmap="magma",annot=True)
plt.show()
```



- Application to Linear Regression

Suppose the data follows linear model $y^{(i)} = h(\vec{x}^{(i)}) + \epsilon^{(i)}$ with unmodeled error ϵ.

Suppose $h(\vec{x}) = \vec{\theta}_*^T \vec{x}$ and the error ϵ follows normal distribution

$$\epsilon \sim \text{Normal}(0, \sigma^2)$$

From the Training Data: $\left(\vec{x}^{(i)}, y^{(i)}\right)$ for $i = 1 \ldots n$.

To minimize the cost function, by normal equation,

$$\hat{\vec{\theta}} = (X^T X)^{-1} X^T \vec{y}$$

Proposition: The covariance matrix of $\vec{\theta}$ can be calculated by

$$\text{Cov}(\vec{\theta}) = \sigma^2 (X^T X)^{-1}$$

Lemma. The covariance matrix of $\vec{z} = A\vec{x}$ is

$$\mathrm{Cov}(\vec{z}) = A\,\mathrm{Cov}(\vec{x})\,A^T$$

Gilbert Strang: "To me, this neat formula shows the beauty of matrix multiplication. I won't prove this formula, just admire it. It is constantly used in applications".

Proof of the Proposition:

$$\begin{aligned}
\mathrm{Cov}(\hat{\vec{\theta}}) &= \mathrm{Cov}((X^TX)^{-1}X^T\vec{y}) \\
&= \mathrm{Cov}((X^TX)^{-1}X^T(X\vec{\theta}_* + \vec{\epsilon})) \\
&= \mathrm{Cov}(\theta_* + (X^TX)^{-1}X^T\vec{\epsilon}) \\
&= \mathrm{Cov}((X^TX)^{-1}X^T\epsilon) \\
&= (X^TX)^{-1}X^T\,\sigma^2 I\,((X^TX)^{-1}X^T)^T \\
&= \sigma^2(X^TX)^{-1}
\end{aligned}$$

lemma

Here, $\vec{\theta}_*$ is the true parameter.

Proof of the Lemma (Exercise)

$$\begin{aligned}
\mathrm{Cov}(\vec{z}) &= E(\vec{z}\vec{z}^T - E(\vec{z})E(\vec{z})^T) \\
&= E(A\vec{x}(A\vec{x})\hat{}T - E(A\vec{x})E(A\vec{x})^T) \\
&= AE(\vec{x}\vec{x}^T - E(\vec{x})E(\vec{x})^T)A^T \\
&= A\,\mathrm{Cov}(\vec{x})\,A^T
\end{aligned}$$

II. We want to understand p-values and confidence intervals of the parameters θ_i

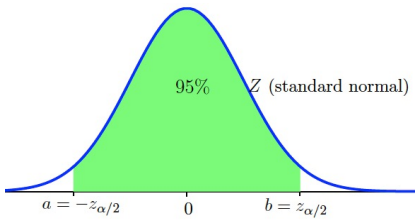
Review:

The **sample mean** of test statistics $x_1 \ldots x_n$ is

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Central Limit Theorem (CLT): Assume that the distribution of test statistics $x_1 \ldots x_n$ is drawn independently from a distribution with mean μ and variance σ^2, then the sample mean follows normal distribution

$$\frac{\bar{x} - \mu}{\sigma/\sqrt{n}} \xrightarrow{n \text{ large}} \mathcal{N}(1, 0)$$



The $\mathbf{1-\alpha}$ **confidence interval** for μ is a set I such that

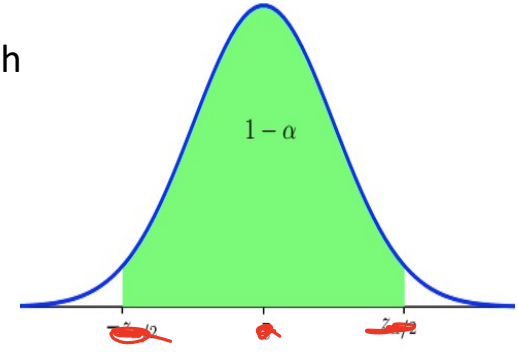
$$P\left(\mu \in \left[\bar{x} - z_{\alpha/2}\left(\frac{\sigma}{\sqrt{n}}\right),\ \bar{x} + z_{\alpha/2}\left(\frac{\sigma}{\sqrt{n}}\right)\right]\right) = 1-\alpha$$

The above explicit calculation for confidence interval is by CLT and the more general definition of confidence interval.

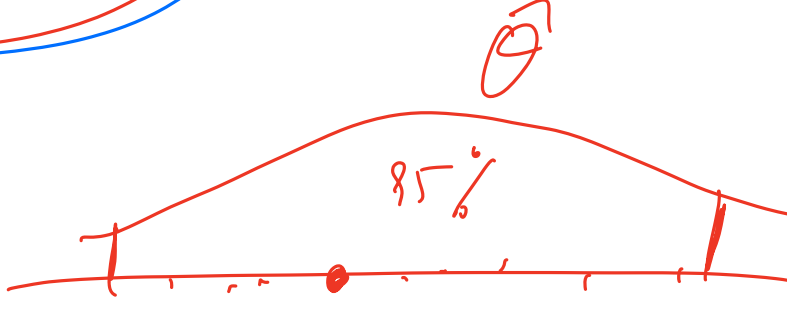
$\theta \sim$ distribution

The $1-\alpha$ confidence interval for any statistic $\hat{\theta}$ is a set I such

$$P_\theta(\theta \in I) \geq 1-\alpha$$



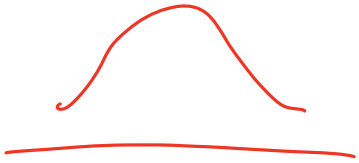
$$\hat{\theta} = (X^T X)^{-1} X^T \vec{y}$$



- Statistical properties of least squares estimate:

Since we assume that error ϵ follows normal distribution $Normal(0, \sigma^2)$, the linear parameters $\hat{\vec{\theta}}$ are normally distributed around the true solution $\vec{\theta}_*$ with covariance matrix $\sigma^2(X^TX)^{-1}$.

$$\hat{\vec{\theta}} \overset{\sim}{\longrightarrow} \text{Normal}(\vec{\theta}_*, \sigma^2(X^TX)^{-1})$$



$$\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \sim \sigma^2 \chi^2_{n-d-1}$$

$$\frac{(\hat{\theta} - \vec{\theta}_*)^T (X^TX)(\hat{\theta} - \vec{\theta}_*)/d+1}{s^2} \sim F_{d+1,n-d-1}$$

$$s^2 = \frac{1}{n-d-1}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \frac{RSS(\theta)}{n-d-1}$$

- Confidence Interval

The $1-\alpha$ **confidence interval** around each θ_i is

$$\left[\hat{\theta}_i - z\alpha/_2\sigma\sqrt{(X^TX)^{-1}_{ii}}\ ,\qquad \hat{\theta}_i + z\alpha/_2\sigma\sqrt{(X^TX)^{-1}_{ii}}\right]$$

- Hypothesis Tests:

To test the hypothesis that $\theta_i = 0$, that is the i-th feature has no bearing on the outcome, we write the **standardized z-Score of θ_i** (test statistic)

$$z_i \approx \frac{\widehat{\theta}_i}{\sigma\sqrt{(X^TX)^{-1}}}$$

- t-distributions and t-tests

We can estimate the variance σ^2 of ϵ from the data by the unbiased estimator:

$$\hat{\sigma}^2 = s^2 = \frac{RSS(\theta)}{n-d-1}$$

Since we need to estimate σ^2 by s^2, the distribution needs to be modified to be t-distribution, and the corresponding CI and test score/statistics will be

$$\left[\hat{\theta} - t_{n-d-1,\alpha/2}\, s\sqrt{(X^TX)^{-1}_{ii}}\ ,\qquad \hat{\theta} + t_{n-d-1,\alpha/2}\, s\sqrt{(X^TX)^{-1}_{ii}}\right]$$

t-Score of θ_i $\qquad t_i \approx \frac{\hat{\theta}_i}{s\sqrt{(X^TX)^{-1}}}$

t-distribution is close to z-distribution when the degree of freedom $n-d-1$ is large.

III (Best) Subset Selection Methods

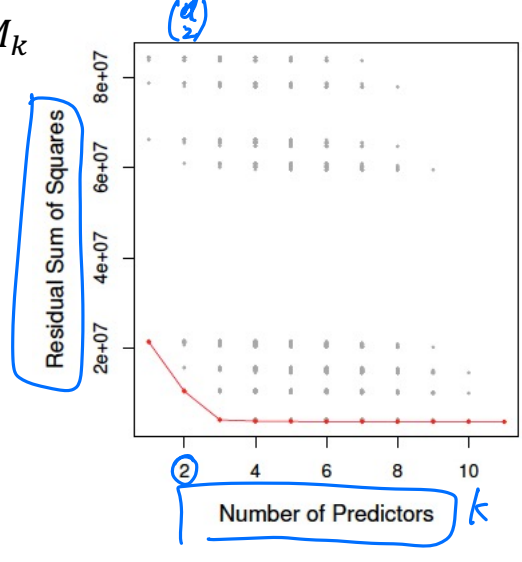
Choose all possible subset combinations of inputs $x_1, \dots, x_d$.

With d variables, there are 2^d many distinct combinations.

Identify the best model among these models.

Algorithm:

1. Let M_0 be the null model, $y = \theta_0 + \epsilon$. The predictor is the sample mean of response.
2. For each $k = 1,2, \dots, d$, fit all $\binom{d}{k}$ models that contain exactly k predictors.
3. Pick the best model that with (smallest RSS) and call it M_k
4. Select a single best model from $M_0, \dots, M_d$ (by RSS?)



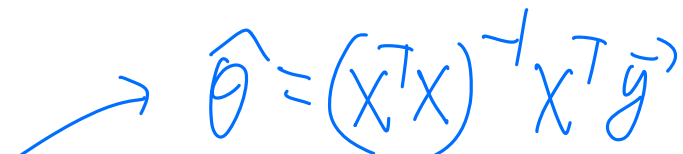
Pros of best subset selection:

1. Straightforward to carry out.
2. Conceptually clear.

Cons of best subset selection

1. The search space too large (2^d models), may lead to overfit.
2. Computationally infeasible: too many models to run.

   For example, if d = 20, there are 2^{20} models.



statsmodels.api/ smf.ols / model.summary()

Dep. Variable:	PRICE	R-squared:	0.84
Model:	OLS	Adj. R-squared:	0.822
Method:	Least Squares	F-statistic:	44.77
Date:	Sat, 23 Jan 2021	Prob (F-statistic):	1.23E-18
Time:	13:52:30	Log-Likelihood:	-817.45
No. Observations:	58	AIC:	1649
Df Residuals:	51	BIC:	1663
Df Model:	6		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-832500.00	2420000.00	-0.35	0.73	-5680000.00	4020000.00
BEDS	27580.00	55400.00	0.50	0.62	-83700.00	139000.00
BATHS	-66690.00	60100.00	-1.11	0.27	-187000.00	53900.00
SQUARE_FEET	426.23	70.62	6.04	0.00	284.45	568.01
LOT_SIZE	22.79	8.79	2.59	0.01	5.14	40.43
YEAR_BUILT	403.80	1240.96	0.33	0.75	-2087.54	2895.14
DAYS_ON_MARKET	2761.00	6061.43	0.46	0.65	-9407.82	14900.00

More error prediction methods.

Mean Square Error $MSE = \frac{RSS}{n}$

$\hat{\theta} = \arg\min_{\theta} RSS(\theta)$

0. R-squared

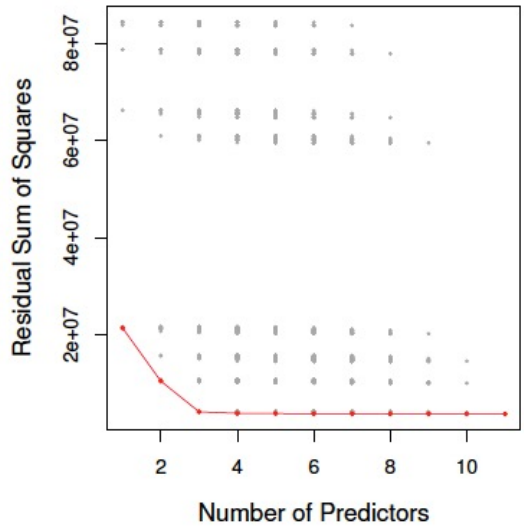
There is human readable scoring statistic is **R-squared** calculated by

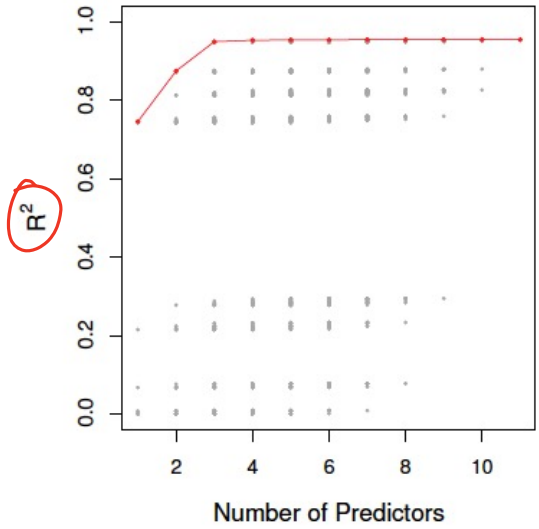
$$R^2 = 1 - \frac{RSS}{SS_{total}} = 1 - \frac{RSS}{\sum_{i=1}^{n}(y^{(i)} - \bar{y})^2}$$

$RSS = \sum_{i=1}^{n}(y^{(i)} - X\theta)^2$, $h_\theta(x)$

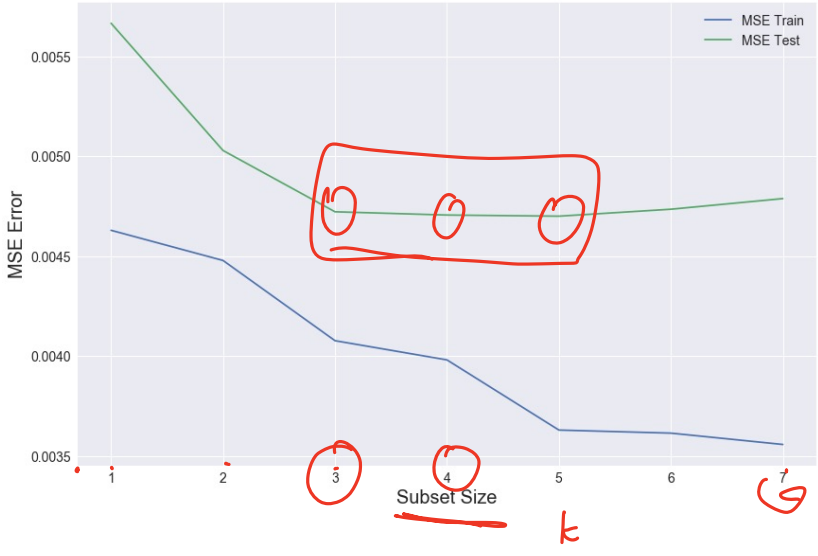
SS_{total} = total S.S = TSS

So $R^2 = 1$ is perfect correlation.





The MSE and R-squared reflects the training error. However, a model with larger R-squared/ or smaller MSE error is not necessarily better than another model with smaller R-squared when we consider test error!



Next we will introduce Adjusted R^2, Mallows' C^p, AIC, BIC for error prediction.

In the following classes, we will also introduce Validation/cross-validation approach.

Suppose we check subset of size k

1. Adjusted R-squared.

The adjusted R-squared, taking into account of the degrees of freedom

$$\textbf{adjusted } R^2 := 1 - \frac{RSS/(n-k-1)}{\sum_{i=1}^{n}(y^{(i)} - \bar{y})^2/(n-1)}$$

With more inputs, the R^2 always increase, but the adjusted R^2 could decrease since more irrelevant inputs are penalized by the smaller degree of freedom of the residuals. The adjusted R-squared is preferred over the R-squared in evaluating models.

2. Mallows' C_p.

The statistic of Mallow's C_p is defined as

$$\text{Mallows' } C_p = \frac{1}{n}(RSS(k) + 2ks_d^2)$$

Here, $s_d^2 = \frac{RSS}{n-d-1}$ and $RSS(k)$ is the RSS with k features.

Mallows' C_p is an unbiased estimate of test MSE. The model with the **smallest** C_p is preferred.

3. Akaike information criterion (AIC)

The goal of AIC is to maximize the predictive likelihood.

$$AIC := \frac{1}{ns_d^2}(RSS(k) + 2ds_d^2)$$

Here, $s_d^2 = RSS/(n-d-1)$

When Gaussian likelihood is assumed in least square regression. The model with the **smallest** AIC is preferred.

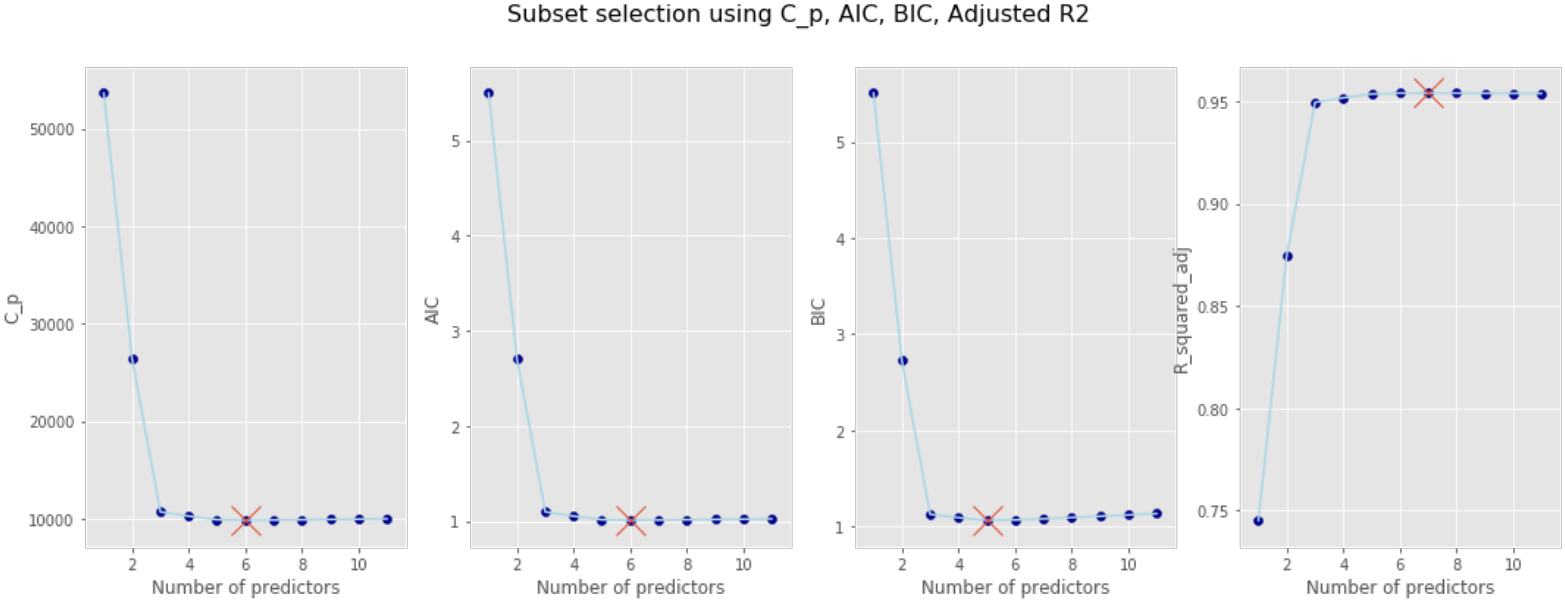
4. Schwarz's Bayesian information criterion (BIC)

For a linear model with d inputs

$$BIC := \frac{1}{ns_d^2}(RSS(k) + ds_d^2(\log n))$$

Again, the model with the **smallest** BIC is preferred.

It replaces $2ds_d^2$ from AIC by $ds_d^2(\log n))$. So, for $\log n > 2$ or $n > 7$, BIC penalizes more heavily the models with more number of inputs.


Subset selection using C_p, AIC, BIC, Adjusted R2
C_p
AIC
BIC
R_squared_adj
Number of predictors
Number of predictors
Number of predictors
Number of predictors

Since the computation of best subset selection is too heavy, we introduce two methods with less computation.

1. **Forward** Subset Selection

The algorithm starts by fitting the intercept θ_0, and then sequentially adds into the model the variable that most improves the fit.

Algorithm:

1. Let M_0 be the null model, $y = \theta_0 + \epsilon$. The predictor is the sample mean of response.
2. For each $k = 0, 1, 2, \ldots, d-1$, consider all $d - k$ models that augment the predictors in M_k with one additional predictor.
3. Pick the best model that with (smallest RSS) and call it M_{k+1}
4. Select a single best model from $M_0, \ldots, M_d$ by AIC or BIC or C_p or adjusted R^2.

Forward Subset Selection has less models (1 + d(d + 1)/2), hence less computation. Once an input is in, it does not get out. No problem for first n-steps if d > n.

2. **Backward** Subset Selection

Backward subset regression starts with the full model and sequentially deletes the predictor with the smallest z-score.

Algorithm:

1. Start with the largest model M_d with all d inputs.
2. For $k = d, d-1, \ldots, 1$, Consider all k models that contain all but one of the predictors in M_k for a total of $k-1$ predictors.
3. Pick the best model that with (smallest RSS) and call it M_{k+1}
4. Select a single best model from $M_0, \ldots, M_d$ by AIC or BIC or C_p or adjusted R^2.

Forward Subset Selection has less models (1 + d(d + 1)/2), hence less computation. Once an input is out, it does not get in. No applicable to the case with d > n

Recall the least squares solution $\vec{\theta} = (X^T X)^{-1} X^T \vec{y}$

The matrix $H = X(X^T X)^{-1} X^T$ is the projection matrix.

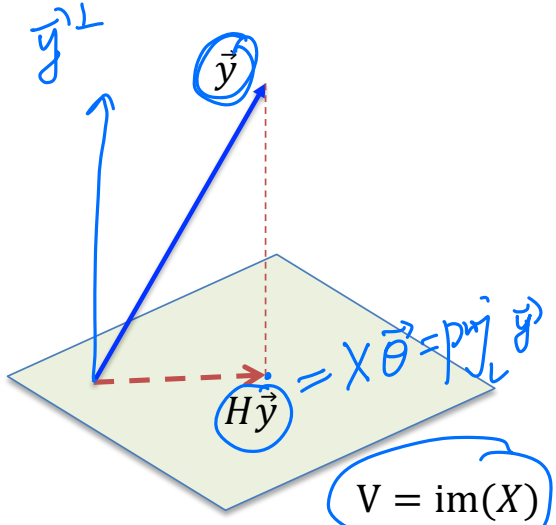
$\hat{y} = X\vec{\theta} = H\vec{y} = \text{Proj}_V \vec{y}$

H is symmetric and idempotent. ($H^T = H$, $H^2 = H$)

Eigenvalues of H are either 1 or 0.

Trace(H) = rank(H)

$$\|\vec{y}\|^2 = \|X\vec{\theta}\|^2 + \|\vec{y} - X\vec{\theta}\|^2$$



Solve $X\vec{\theta} = \vec{y}$

no soln.

Find $\vec{\theta}$ s.t.

$X\vec{\theta} = \text{proj}_V \vec{y}$

IV. Gauss-Markov Theorem.

Suppose the data follows linear model $y = \vec{\theta}_*^T \vec{x} + \epsilon$ with error $\epsilon \sim \text{Normal}(0, \sigma^2)$

Gauss-Markov Theorem. Among all *linear unbiased* estimates for the solution to $X\hat{\theta} = \vec{y}$, the least squares estimate $\hat{\theta} = (X^T X)^{-1} X^T \vec{y}$ has the *smallest variance*.

Proof: Suppose $\vec{\theta}' = A\vec{y}$ is another unbiased **linear** estimate.

By unbiasedness, $E(A\vec{y}) = \vec{\theta}_*$, that is $E\left(A\left(X\vec{\theta}_* + \epsilon I\right)\right) = \vec{\theta}_*$. Hence $AX = I$

By Lemma, $\text{Cov}(\vec{\theta}') = A\ \text{Cov}(\vec{y})\ A^T = \sigma^2 AA^T$

By Proposition, $\text{Cov}(\hat{\theta}) = \sigma^2 (X^T X)^{-1}$

Let $D = A - (X^T X)^{-1} X^T$, then $DX = 0$.

Then $AA^T - (X^T X)^{-1} = DD^T$ which is positive semidefinite.

Remarks:

The Gauss-Markov Theorem implies that the least squares estimator has the smallest *mean squared error* of any *unbiased linear* estimator.

Least squares estimator is the Best Linear Unbiased Estimator (BLUE).

There may still exists biased estimators with a smaller mean squared error. That is, we may be able to **trade** a small increase in bias for a large reduction in variance.

We already learned Ridge/ Lasso/ Elastic net regression methods for regularization.

Subset selection method provides another way of doing this. If your selection procedure drops coefficients whose true value is nonzero, you will incur an error due to bias. However, subset selection is a discrete process and so often exhibits high variance.

➢ Feature/variable selection

Not all existing input variables are useful for predicting the output. Keeping redundant inputs in model can lead to poor prediction and poor interpretation.

1. Prediction accuracy: Least squares estimates often have low bias at the cost of high variance. Prediction accuracy can sometimes be improved by shrinking some coefficients to zero. (reduce variance)

2. Interpretation power: With a large number of related parameters, one often wants to find a small number of predictors with strongest effects.

We already learned Shrinkage/regularization methods to constrain some regression parameters to 0.
We also learned Best Subset Selection Methods.
Later, we also have dimension reduction method for high dimension data analysis.

Useful libraries of Python:

1. **NumPy**: For large, multi-dimensional arrays and matrices.
2. **pandas**: data manipulation and analysis.
3. **Matplotlib**: Visualization with Python.
4. **seaborn**: statistical data visualization visualization based on Matplotlib.
5. **scikit-learn**: various classification, regression and clustering algorithms.
6. **Statsmodels**: classes and functions for the estimation of statistical models.
7. …

Libraries for deep learning:

1. **TensorFlow**: focus on training and inference of deep neural networks.
2. **Keras**: interface for the TensorFlow.
3. **Pytorch**: focus on computer vision and natural language processing.
4. **OpenCV**: real-time computer vision.