

# Feature Extraction and Cross Validation

## Supervised Linear Regression

# Agenda

- Interaction Effect & Dummy Variable
- Machine Learning Pipeline
- Data Processing
- Feature Extraction
  - Feature Transformation
    - Logarithmic transformation
    - Square root transformation
    - Reciprocal transformation
    - Exponential transformation
    - Box-cox transformation

# Agenda

- Feature Scaling
  - Normalization
  - Standardization
- Feature Selection
  - Forward selection method
  - Backward elimination method
  - Stepwise method
  - Recursive Feature Elimination

# Agenda

- Optimization
  - Prediction Evaluation
    - Bias and Variance
  - Model Validation
    - K-Fold Cross Validation
    - LOOCV

# Interaction Effect

# Interaction effect

Sentiment

 +		=	Salt water			
		=	Sweet water			
		=	Lemon water			
	  	=	Lemonade			

# Interaction

- An interaction effect occurs when the effect of one variable depends on another variable. This combined effect may or may not improve the performance of the model
- Note: It does not imply that the predictor variables are collinear

Example: Salary of an employee increases with experience, but this may vary based on whether the person has completed additional courses like MBA

# Interaction Effect

- In context with our example, we shall consider the interaction effect of variables Engine\_Capacity and Mileage
- We obtained Int\_EC\_Mil by taking the product of Mileage and Engine\_Capacity
- Let us check whether the interaction term is adding value to our model

Mileage	Engine_Capacity	Int_EC_Mil	Age	Premium (in dollars)
15	1.8	27	2	392.5
14	1.2	16.8	10	46.2
17	1.2	20.4	8	15.7
7	1.8	12.6	3	422.2
10	1.6	16	4	119.4
7	1.4	9.8	3	170.9
20	1.2	24	7	56.9
21	1.6	33.6	6	77.5
18	1.2	21.6	2	214
11	1.6	17.6	5	65.3
7.9	1.4	11.06	3	250
8.6	1.6	13.76	3	220
12.3	1.2	14.76	2	217.5
17.1	1.6	27.36	1	140.88
19.4	1.2	23.28	6	97.25



# Interaction Effect



Now our model is:

$$\text{Premium} = \beta_0 + \beta_1 \text{ Mileage} + \beta_2 \text{ Engine\_Capacity} + \beta_3 \text{ Age} + \beta_4 \text{ Int\_EC\_Mil} + \varepsilon$$

Parameter	Description
$\beta_0$	Premium value where the best fit line cuts the Y-axis (Premium)
$\beta_1$	Regression coefficient of the variable Mileage
$\beta_2$	Regression coefficient of the variable Engine_Capacity
$\beta_3$	Regression coefficient of the variable Age
$\beta_4$	Regression coefficient of the variable Int_EC_Mil

Dep. Variable:	Premium	R-squared:	0.645			
Model:	OLS	Adj. R-squared:	0.503			
Method:	Least Squares	F-statistic:	4.543			
Date:	Sat, 02 Jan 2021	Prob (F-statistic):	0.0238			
Time:	21:50:39	Log-Likelihood:	-84.981			
No. Observations:	15	AIC:	180.0			
Df Residuals:	10	BIC:	183.5			
Df Model:	4					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-502.0111	563.770	-0.890	0.394	-1758.169	754.147
Mileage	40.3059	37.155	1.085	0.303	-42.480	123.092
Engine	568.7234	369.858	1.538	0.155	-255.371	1392.817
Age	-25.7814	10.348	-2.491	0.032	-48.838	-2.724
Int_Engine_Mil	-30.5470	24.879	-1.228	0.248	-85.981	24.887
Omnibus:	0.026	Durbin-Watson:	2.101			
Prob(Omnibus):	0.987	Jarque-Bera (JB):	0.195			
Skew:	0.079	Prob(JB):	0.907			
Kurtosis:	2.464	Cond. No.	774.			

# Linear regression model (interaction effect)



Based on the data, the  $\beta$  parameters are:

$$\beta_0 = -502.011, \beta_1 = 40.3059, \beta_2 = 568.723,$$

$$\beta_3 = -25.7814 \text{ \& } \beta_4 = -30.5470$$

Thus the model is

$$Y = 502.011 + 40.3059 x_1 + 568.723 x_2 - 25.7814 x_3 - 30.5470 x_4$$

That is,

$$\text{Premium} = -502.011 + 40.3059 * \text{Mileage} + 568.723 *$$

$$\text{Engine\_Capacity} - 25.7814 * \text{Age} - 30.5470 * \text{Int\_EC\_Mil}$$

Dep. Variable:	Premium	R-squared:	0.645			
Model:	OLS	Adj. R-squared:	0.503			
Method:	Least Squares	F-statistic:	4.543			
Date:	Sat, 02 Jan 2021	Prob (F-statistic):	0.0238			
Time:	21:50:39	Log-Likelihood:	-84.981			
No. Observations:	15	AIC:	180.0			
Df Residuals:	10	BIC:	183.5			
Df Model:	4					
Covariance Type:	nonrobust					
	coef	std err	t	P> t	[0.025	0.975]
Intercept	-502.011	563.770	-0.890	0.394	-1758.169	754.147
Mileage	40.3059	37.155	1.085	0.303	-42.480	123.092
Engine	568.7234	369.858	1.538	0.155	-255.371	1392.817
Age	-25.7814	10.348	-2.491	0.032	-48.838	-2.724
Int_Engine_Mil	-30.5470	24.879	-1.228	0.248	-85.981	24.887
Omnibus:	0.026	Durbin-Watson:	2.101			
Prob(Omnibus):	0.987	Jarque-Bera (JB):	0.195			
Skew:	0.079	Prob(JB):	0.907			
Kurtosis:	2.464	Cond. No.	774.			

Inference: By Looking at the P-value of the Interaction Effect of Engine & Mileage, we can say that there is no Interaction between Mileage & Engine that produces significant result for the premium calculation.

Presence of categorical variable

# Linear regression of categorical variable

- The regression method fails in presence of categorical variable
- Thus we need to convert the categorical variable to numeric variable
- In order to so, we use  $N - 1$  dummy encoding

# N-1 dummy encoding

- Dummy variables are binary variables used to represent categorical data
- For a categorical variable that can take  $k$  values,  $k-1$  dummy variables need to be created
- A dummy variable is 1 if it takes a particular value, else it is 0

# Dummy variable example

Consider a variable, Gender, used to represent the gender of a citizen during the census

Gender: Male, Female

Since Gender takes 2 values it can be represented with 1 dummy variable  $D_1$  as:

Value	$D_1$
Male	0
Female	1

# Data

Let us consider a categorical variable Manufacturer in the data and find out how it behaves.

Mileage	Manufacturer	Premium (in dollars)
15	Ford	392.5
14	Honda	46.2
17	Tata	15.7
7	Ford	422.2
10	Ford	119.4
7	Tata	170.9
20	Tata	56.9
21	Honda	77.5
18	Honda	214
11	Tata	65.3
7.9	Ford	250
8.6	Tata	220
12.3	Tata	217.5
17.1	Ford	140.88
19.4	Honda	97.25

# Example

- In context with our example, the categorical variable Manufacturer takes values Ford, Honda and Tata
- Since Manufacturer takes 3 values, two dummy variables Mfr\_Honda and Mfr\_Tata are created

Value	Mfr_Honda	Mfr_Tata
Ford	0	0
Honda	1	0
Tata	0	1



# Model with categorical variable

Now our model is

$$\text{Premium} = \beta_0 + \beta_1 \text{ Mileage} + \beta_2 \text{ Mfr\_Honda} + \beta_3 \text{ Mfr\_Tata} + \varepsilon$$

Parameter	Description
$\beta_0$	Premium value where the best fit line cuts the Y-axis (Premium)
$\beta_1$	Regression coefficient of the variable Mileage
$\beta_2$	Regression coefficient of the dummy variable Mfr_Honda
$\beta_3$	Regression coefficient of the dummy variable Mfr_Tata

# Linear regression model (dummy variable)

Based on the data, the  $\beta$  parameters are:

$$\beta_0 = 368.93, \beta_1 = -9.117,$$

$$\beta_2 = -95.174 \text{ and } \beta_3 = -129.216$$

Thus the model is

$$Y = 368.93 - 9.117 x_1 - 95.174 x_2 - 129.216 x_3$$

That is,

$$\text{Premium} = 368.93 - 9.117 \text{ Mileage} - 95.174 \text{ Mfr\_Honda} - 129.216 \text{ Mfr\_Tata}$$

Mileage	Manufacturer	Premium (in dollars)
15	Ford	392.5
14	Honda	46.2
17	Tata	15.7
7	Ford	422.2
10	Ford	119.4
7	Tata	170.9
20	Tata	56.9
21	Honda	77.5
18	Honda	214
11	Tata	65.3
7.9	Ford	250
8.6	Tata	220
12.3	Tata	217.5
17.1	Ford	140.88
19.4	Honda	97.25

# Regression line (dummy variable)

The regression line:

$$\text{Premium} = \beta_0 + \beta_1 \text{ Mileage} + \beta_2 \text{ Mfr\_Honda} + \beta_3 \text{ Mfr\_Tata} + \varepsilon$$

If the manufacturer is Honda, the regression line becomes:

$$\begin{aligned} \text{Premium} &= \beta_0 + \beta_1 \text{ Mileage} + \beta_2 \text{ Mfr\_Honda} + \beta_3 \text{ Mfr\_Tata} \\ &= \beta_0 + \beta_1 \text{ Mileage} + \beta_2 (1) + \beta_3 (0) \\ &= \beta_0 + \beta_1 \text{ Mileage} + \beta_2 + 0 \\ &= (\beta_0 + \beta_2) + \beta_1 \text{ Mileage} \end{aligned}$$

Value	Mfr_Honda	Mfr_Tata
Ford	0	0
Honda	1	0
Tata	0	1

Note the change in the intercept value.

# Regression line (dummy variable)

The regression line:

$$\text{Premium} = \beta_0 + \beta_1 \text{ Mileage} + \beta_2 \text{ Mfr\_Honda} + \beta_3 \text{ Mfr\_Tata} + \varepsilon$$

Value	Mfr_Honda	Mfr_Tata
Ford	0	0
Honda	1	0
Tata	0	1

For manufacturer = Ford,

$$\text{Premium} = \beta_0 + \beta_1 \text{ Mileage}$$

Actual intercept

For manufacturer = Honda,

$$\text{Premium} = (\beta_0 + \beta_2) + \beta_1 \text{ Mileage}$$

Change in intercept

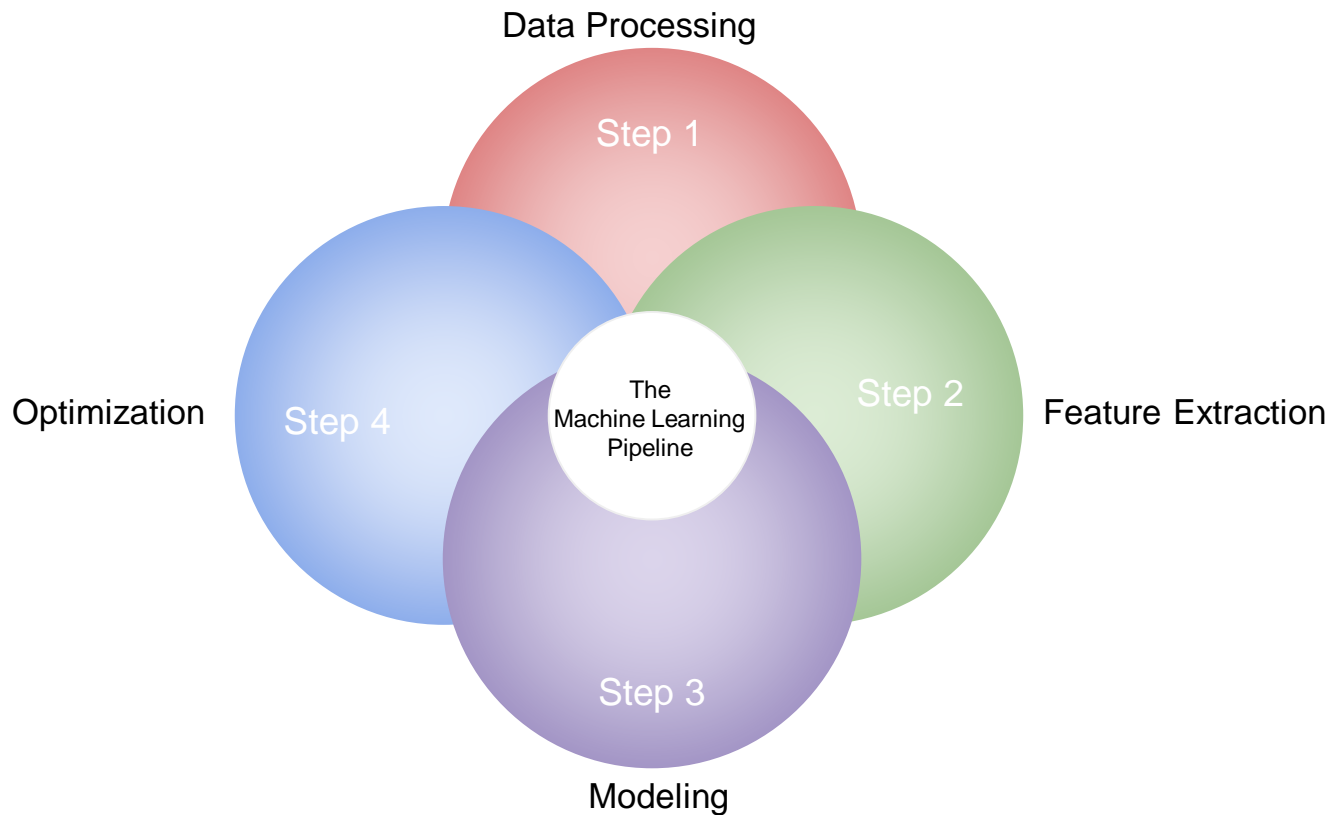
For manufacturer = Tata,

$$\text{Premium} = (\beta_0 + \beta_3) + \beta_1 \text{ Mileage}$$

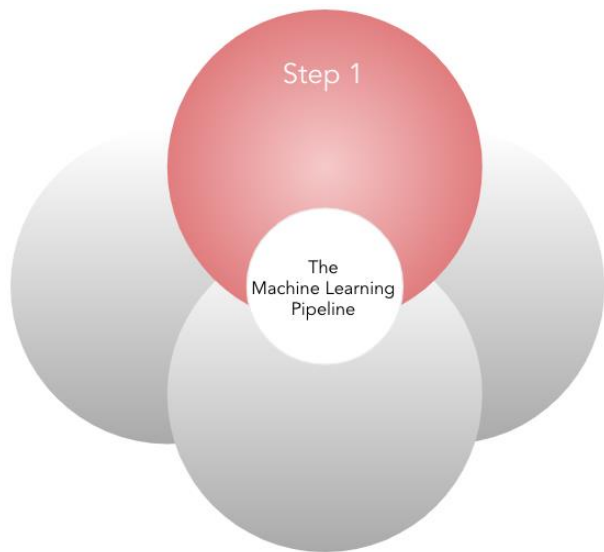
Change in intercept

# Machine Learning Pipeline

# The ML pipeline



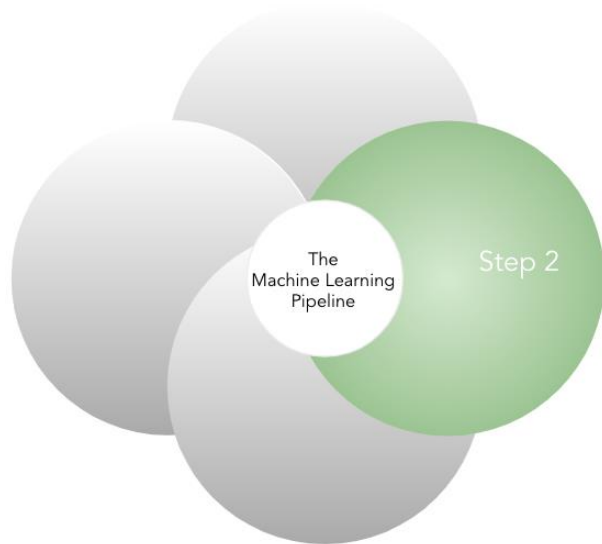
# The ML pipeline: Data processing



## DATA PROCESSING

- Collection
- Formatting
- Labelling

# The ML pipeline: Feature extraction

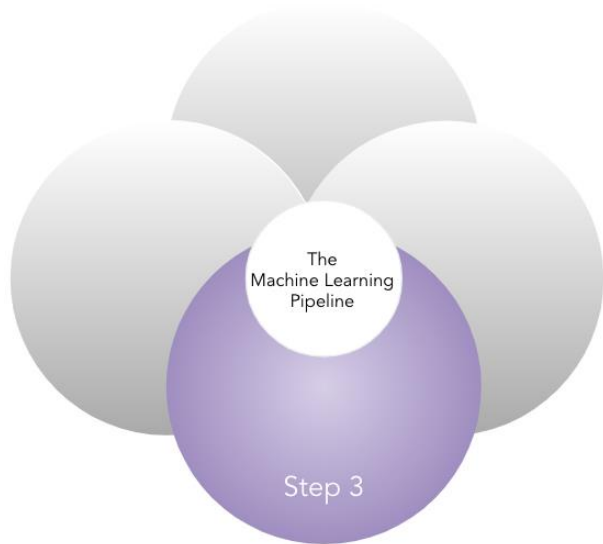


## FEATURE EXTRACTION

- Feature Transformation
- Feature Engineering
- Feature Selection



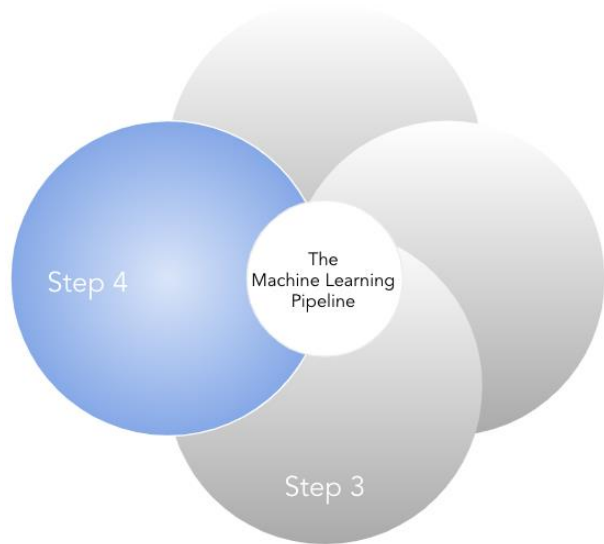
# The ML pipeline: Modeling



## MODELING

- Model Building
- Model Evaluation

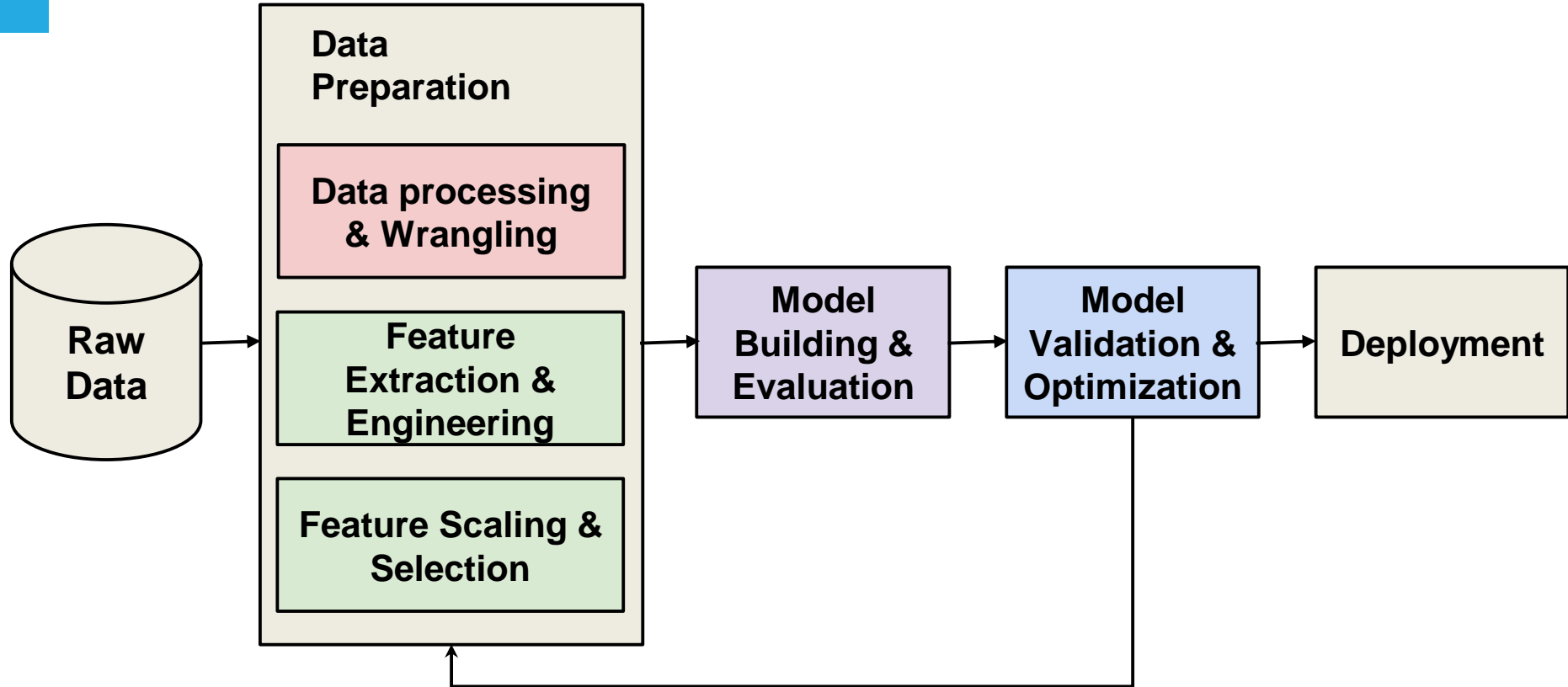
# The ML pipeline: Optimization



## OPTIMIZATION

- Prediction Evaluation
- Model Validation
- Fine Tuning

# The ML pipeline



# Data Processing

# Data processing

## DATA PROCESSING

- Collection
- Formatting
- Labelling


- Collection: To extract data from various sources. Generally obtained in the raw form and not immediately suitable for analysis
- Formatting: Organizing the datasets as required for analysis
- Labelling: Manually labelling data

# Feature Extraction

# Feature

- Feature or attribute is an independent variable that acts as input to our model
- The columns of a dataset are considered as features

Features



Product ID	Store	City
FD_234	A	Chennai
DR_543	A	Bangalore
FD_176	B	Mumbai
DR_621	A	New Delhi

# Feature Extraction

## FEATURE EXTRACTION

- Feature Transformation
  - Feature Engineering
  - Feature Selection
- Feature Transformation: Replacing the existing features by a function of these features
  - Feature Engineering: Creating new features based on empirical relationships
  - Feature Selection: Fitting a model of significant features



# Feature Transformation

# Why do we need feature transformation?

- In case of **skewed (predictor and/or dependent) variable**, we **transform** it to reduce the skewness
- If the assumptions of linear regression are not met, the transformation of skewed target variable can be used for making the error terms more compatible with the assumptions
- If the relationship between a predictor and the response variable is non-linear, it can be linearized using transformation



## Assumption of normality

The parametric methods used to compute test statistics or confidence intervals on the predictor variables assume the data to follow a normal distribution

Hence it is favourable that features have an approximately normal distribution

Recap: The parametric methods are used when sample statistics adequately represent the population

# Rule for transformed variables

Comparison of model performance should be done using the original units for the target variable and not with the units after the transformation

# Transformation methods

- Logarithmic transformation
- Square root transformation
- Reciprocal transformation
- Exponential transformation
- Box-cox transformation

# Transformation methods

- Logarithmic transformation
- Square root transformation
- Reciprocal transformation
- Exponential transformation
- Box-cox transformation

# Logarithmic transformation

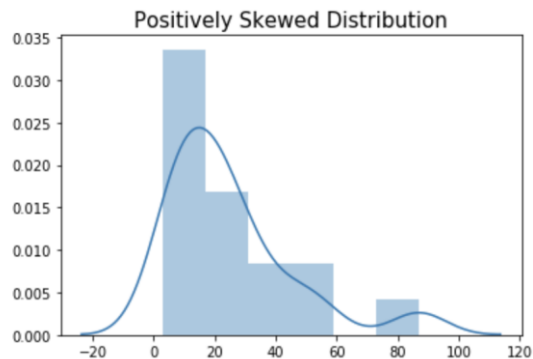
- To linearize, values of a variable are replaced with its natural log
- It cannot be used on a categorical variable after dummy encoding since  $\ln(0)$  is undefined
- Also if a variable takes zero or negative values, the logarithmic transformation cannot be used on it

# Example of log transformation

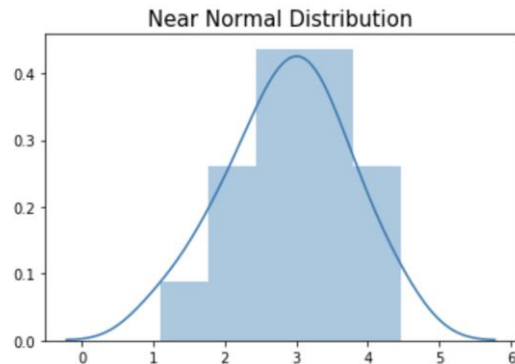
Consider the following data:

Note: Values are rounded to 1 decimals

X	12	9	3	6	24	13	21	6	16	13	54	23	46	32	87	23	34
$\ln(X)$	2.5	2.2	1.1	1.8	3.2	2.6	3.1	1.8	2.7	2.6	3.9	3.1	3.8	3.4	4.5	3.1	3.5



Logarithmic  
Transformation





# Transformation techniques

- Logarithmic transformation
- Square root transformation
- Reciprocal transformation
- Exponential transformation
- Box-cox transformation

# Square root transformation

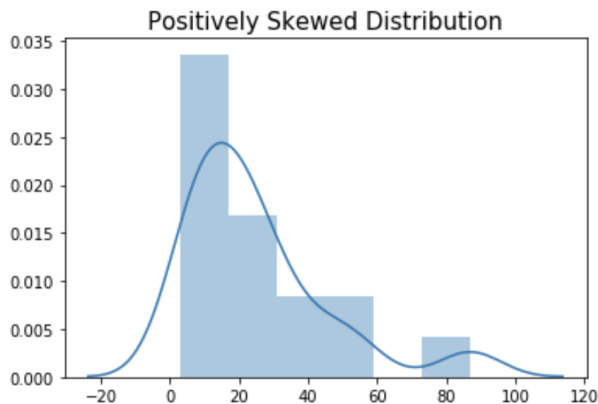
- Values of a variable are replaced with its square root
- To reduce right skewness, we may use square root transformation
- It can be applied even when the variable takes a zero value

# Example of square root transformation

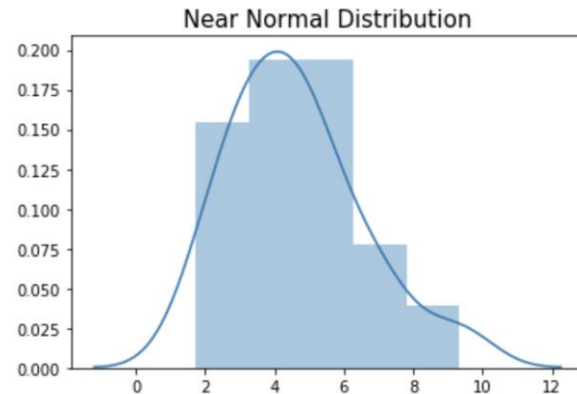
Consider the following data:

Note: Values are rounded to 1 decimals

X	12	9	3	6	24	13	21	6	16	13	54	23	46	32	87	23	34
$\sqrt{X}$	3.5	3	1.7	2.4	4.9	3.6	4.6	2.4	4	3.6	7.4	4.8	6.8	5.7	9.3	4.8	5.8



Square Root Transformation



# Transformation techniques

- Logarithmic transformation
- Square root transformation
- Reciprocal transformation
- Exponential transformation
- Box-cox transformation

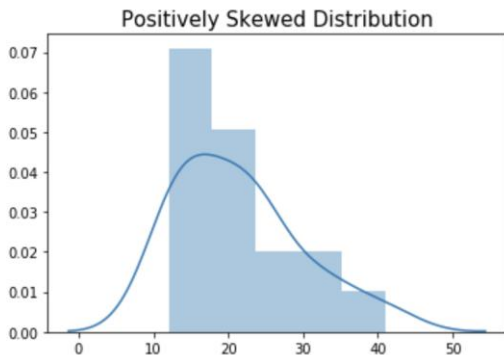
# Reciprocal transformation

- Values of a variable are replaced with its reciprocal
- It can not be applied when the variable takes zero values
- However, can be applied to negative values
- Example: population per area (population density) transforms to area per person

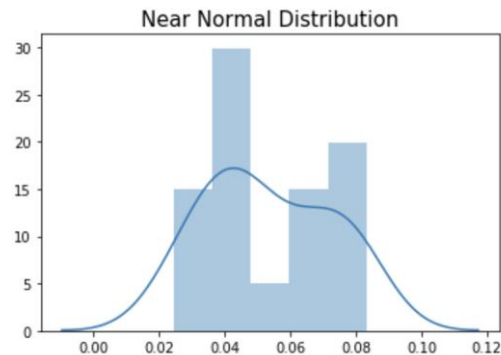
# Example of reciprocal

Consider the following data :

X	12	19	23	16	14	13	21	13	16	13	24	23	41	32	27	23	34
1/X	.08	.05	.04	.06	.07	.08	.05	.08	.06	.08	.04	.04	.02	.03	.04	.04	.03



Reciprocal  
Transformation



# Transformation techniques

- Logarithmic transformation
- Square root transformation
- Reciprocal transformation
- Exponential transformation
- Box-cox transformation

# Exponential transformation

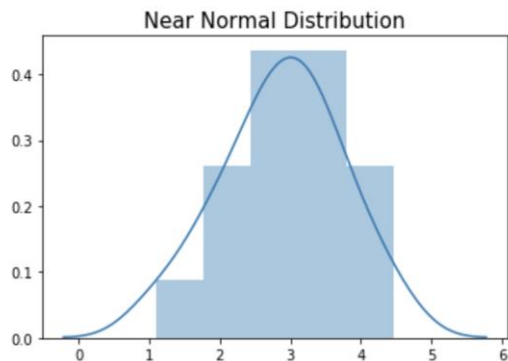
- Values of a variable are replaced with its exponential
- It is generally used to transform logarithmic transformed data to get the original data back



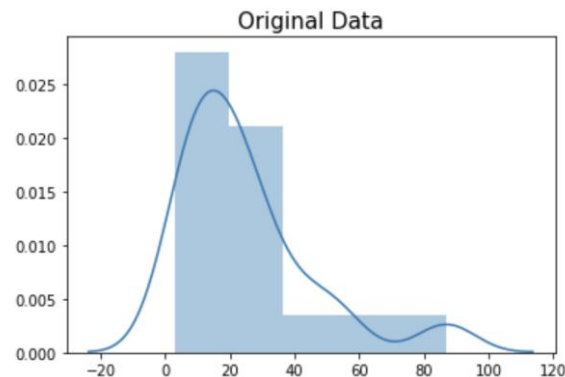
# Example of exponential transformation

Consider the data used in logarithmic transformation.

X	12	9	3	6	24	13	21	6	16	13	54	23	46	32	87	23	34
$\ln(X)$	2.5	2.2	1.1	1.8	3.2	2.6	3.1	1.8	2.7	2.6	3.9	3.1	3.8	3.4	4.5	3.1	3.5
$\exp(X)$	12	9	3	6	24	13	21	6	16	13	54	23	46	32	87	23	34



Exponential  
Transformation



# Transformation techniques

- Logarithmic transformation
- Square root transformation
- Reciprocal transformation
- Exponential transformation
- Box-cox transformation

# Box cox transformation

- It is defined as

$$X^\lambda = \begin{cases} \frac{X^\lambda - 1}{\lambda} & \text{if } \lambda > 0 \\ \ln(X) & \text{if } \lambda = 0 \end{cases}$$

Here,  $X$  is the variable and  $\lambda$  is the transformation parameter and can be tuned according to the data.

- The Box-Cox transformation can only be used on positive variables
- Generalized form of logarithmic transformation

# Summary

Type of Transformation	Properties
Logarithmic transformation	Can not be used if a variable takes zero or negative values
Square root transformation	Can be used to reduce the right skewness, Can not be used if the variable takes negative values
Reciprocal transformation	Can not be used if a variable takes zero value
Exponential transformation	Inverse of log transformation
Box-cox transformation	Generalization of log transformation

# Model Before Log Transformation



Build the multiple linear regression model using the OLS method.

```
# build a model on training dataset
# fit() is used to fit the OLS model
MLR_model = sm.OLS(y_train, X_train).fit()

# print the summary output
print(MLR_model.summary())
```

```
=====
                        OLS Regression Results
=====
Dep. Variable:          Appliances    R-squared:                0.352
Model:                  OLS          Adj. R-squared:           0.315
Method:                 Least Squares  F-statistic:             9.663
Date:                   Mon, 02 Nov 2020  Prob (F-statistic):      1.09e-26
Time:                   16:39:10      Log-Likelihood:          -1942.0
No. Observations:       434          AIC:                     3932.
Df Residuals:           410          BIC:                     4030.
Df Model:                23
Covariance Type:        nonrobust
=====
```

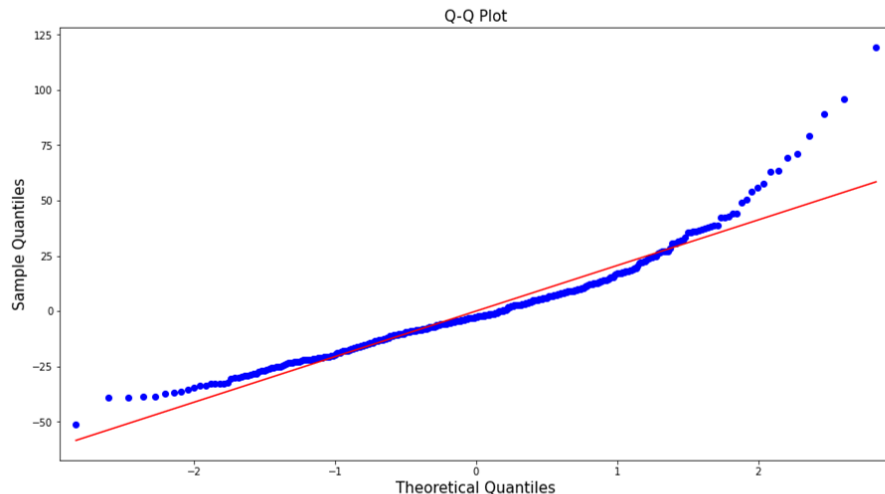
	coef	std err	t	P> t	[0.025	0.975]
const	30.5038	157.184	0.194	0.846	-278.484	339.492
T1	1.6850	3.239	0.520	0.603	-4.683	8.053
RH_1	5.7350	1.404	4.084	0.000	2.974	8.496
T2	-7.0984	2.915	-2.435	0.015	-12.829	-1.368
RH_2	-4.4119	1.387	-3.181	0.002	-7.139	-1.685
T3	4.7152	1.865	2.528	0.012	1.049	8.382
RH_3	0.0217	1.275	0.017	0.986	-2.485	2.528
T4	3.2258	2.027	1.591	0.112	-0.759	7.210
RH_4	1.0482	1.182	0.887	0.376	-1.275	3.372
T5	0.2366	2.129	0.111	0.912	-3.949	4.422
RH_5	-0.0046	0.275	-0.017	0.987	-0.546	0.537
T6	1.4891	0.949	1.569	0.117	-0.377	3.355
RH_6	0.0480	0.107	0.447	0.655	-0.163	0.259
T7	-4.4128	2.288	-1.928	0.055	-8.911	0.086
RH_7	0.5859	0.676	0.867	0.387	-0.743	1.915
T8	11.1364	1.670	6.667	0.000	7.853	14.420
RH_8	-2.9775	0.588	-5.061	0.000	-4.134	-1.821
T9	-8.3401	3.044	-2.740	0.006	-14.324	-2.356
RH_9	-0.8892	0.663	-1.341	0.181	-2.192	0.414
T_out	-0.5231	1.100	-0.476	0.634	-2.685	1.638
Press_mm_hg	0.0053	0.191	0.028	0.978	-0.371	0.381
RH_out	0.0488	0.192	0.255	0.799	-0.328	0.426
Windspeed	1.1656	0.592	1.969	0.050	0.002	2.329
Visibility	-0.0091	0.129	-0.071	0.943	-0.262	0.244

```
=====
Omnibus:                128.702    Durbin-Watson:           1.935
Prob(Omnibus):           0.000     Jarque-Bera (JB):        428.410
Skew:                    1.340     Prob(JB):                9.37e-94
Kurtosis:                7.063     Cond. No.:               1.16e+05
=====
```

```
# plot the Q-Q plot
# 'r' represents the regression line
qqplot(MLR_model.resid, line = 'r')

# set plot and axes labels
# set text size using 'fontsize'
plt.title('Q-Q Plot', fontsize = 15)
plt.xlabel('Theoretical Quantiles', fontsize = 15)
plt.ylabel('Sample Quantiles', fontsize = 15)

# display the plot
plt.show()
```



**Interpretation** Here we can see that the residuals are not normally distributed. The value of skewness is 1.3445. We will log transform the target variable and see if this reduces the skewness.

# Model After Log Transformation



Build the multiple linear regression model using the OLS method after transforming the target variable.

```
# build a model on training dataset
# fit() is used to fit the OLS model
# use log transformation of y_train
MLR_model_after_transform = sm.OLS(np.log(y_train), X_train).fit()

# print the summary output
print(MLR_model_after_transform.summary())
```

```
=====
                        OLS Regression Results
=====
Dep. Variable:          Appliances    R-squared:                0.353
Model:                  OLS          Adj. R-squared:           0.317
Method:                 Least Squares  F-statistic:              9.733
Date:                  Mon, 02 Nov 2020  Prob (F-statistic):      6.79e-27
Time:                  16:39:10       Log-Likelihood:          -114.07
No. Observations:      434           AIC:                     276.1
Df Residuals:          410           BIC:                     373.9
Df Model:               23
Covariance Type:       nonrobust
=====
```

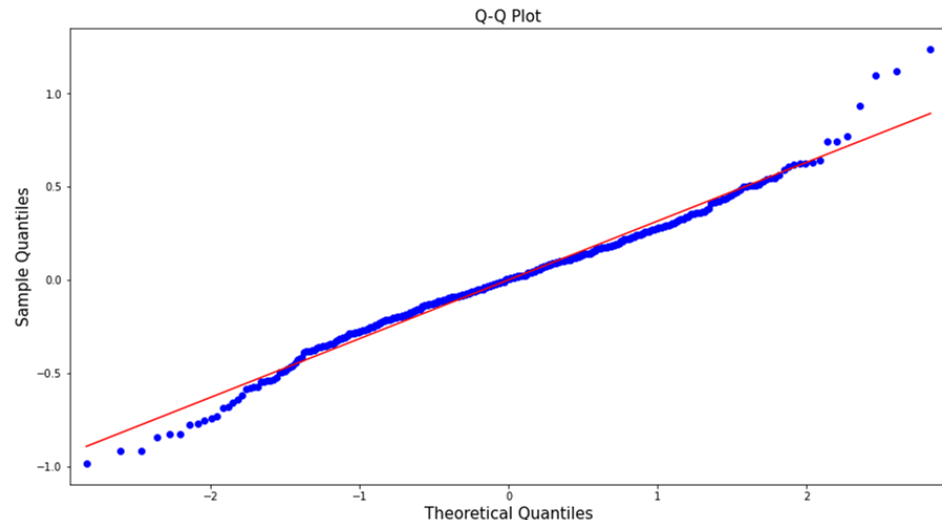
	coef	std err	t	P> t	[0.025	0.975]
const	3.9872	2.330	1.712	0.088	-0.592	8.567
T1	0.0063	0.048	0.131	0.896	-0.088	0.101
RH_1	0.0741	0.021	3.561	0.000	0.033	0.115
T2	-0.0743	0.043	-1.720	0.086	-0.159	0.011
RH_2	-0.0523	0.021	-2.542	0.011	-0.093	-0.012
T3	0.0627	0.028	2.269	0.024	0.008	0.117
RH_3	-0.0043	0.019	-0.228	0.820	-0.041	0.033
T4	0.0586	0.030	1.951	0.052	-0.000	0.118
RH_4	0.0127	0.018	0.725	0.469	-0.022	0.047
T5	-0.0144	0.032	-0.457	0.648	-0.076	0.048
RH_5	0.0010	0.004	0.252	0.801	-0.007	0.009
T6	0.0205	0.014	1.456	0.146	-0.007	0.048
RH_6	0.0004	0.002	0.221	0.825	-0.003	0.003
T7	-0.0621	0.034	-1.831	0.068	-0.129	0.005
RH_7	0.0095	0.010	0.944	0.345	-0.010	0.029
T8	0.1656	0.025	6.688	0.000	0.117	0.214
RH_8	-0.0411	0.009	-4.719	0.000	-0.058	-0.024
T9	-0.1033	0.045	-2.290	0.023	-0.192	-0.015
RH_9	-0.0099	0.010	-1.004	0.316	-0.029	0.009
T_out	-0.0148	0.016	-0.911	0.363	-0.047	0.017
Press_mm_hg	-0.0010	0.003	-0.343	0.732	-0.007	0.005
RH_out	-0.0004	0.003	-0.133	0.894	-0.006	0.005
windspeed	0.0162	0.009	1.844	0.066	-0.001	0.033
Visibility	-0.0001	0.002	-0.054	0.957	-0.004	0.004

```
=====
Omnibus:                14.420    Durbin-Watson:              1.970
Prob(Omnibus):           0.001    Jarque-Bera (JB):          31.139
Skew:                    0.037    Prob(JB):                  1.73e-07
Kurtosis:                4.310    Cond. No.                  1.16e+05
=====
```

```
# plot the Q-Q plot
# 'r' represents the regression line
qqplot(MLR_model_after_transform.resid, line = 'r')

# set plot and axes labels
# set text size using 'fontsize'
plt.title('Q-Q Plot', fontsize = 15)
plt.xlabel('Theoretical Quantiles', fontsize = 15)
plt.ylabel('Sample Quantiles', fontsize = 15)

# display the plot
plt.show()
```



After transforming the target variable, the skewness reduced to 0.0366 and we can see a near normal distribution of the residuals.

# Feature Scaling

# Feature scaling

- It is a technique used to transform the data into a same scale
- Since the features have various ranges, it becomes a necessary step in data preprocessing while using machine learning algorithms
- Since most machine learning algorithms use distance calculations, features taking higher values will weigh in more in the distance compared to features taking values of low magnitude



# Example

- In a dataset which has variables age and income. The age of a person is measured in years which can take values between 18 to 65 (retirement age) and income of a person is in thousands  
So it is necessary to bring the two features on the same scale to assign appropriate weights
- In some parts of the world, height is measured using metric system (centimetres), while in some other parts the imperial system is used (feet/inches).  
So the results would be different if the height value is 152 cm or 5 feet, when if converted they refer to the same height value.

# Feature scaling methods

- Normalization
- Standardization

# Normalization

Normalization is the process of rescaling features in the range 0 to 1

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

# Standardization

- Standardization rescales the feature such that it has mean 0 and unit variance
- The procedure involves subtracting the mean from observation and then dividing by the standard deviation

$$x' = \frac{x - \bar{x}}{\sigma}$$

# Feature Selection

# Feature selection

- Feature selection is the process of including the significant features in the model
- This can be achieved by:
  - Forward selection method
  - Backward elimination method
  - Stepwise method
- To understand the above methods let  $X_1, X_2, \dots, X_k$  be  $k$  predictor variables and  $Y$  be the response variable

# Forward selection method

## Procedure

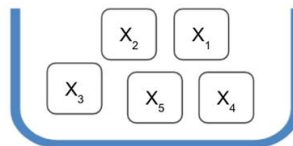
1. Start with a null model (with no predictors)
2. Obtain the correlation between  $Y$  and each variable. The variable with highest correlation gets added to the model (say  $X_m$ ). Build a model  $Y \sim X_m$
3. Obtain the correlation between  $Y$  and remaining  $(k-1)$  variables. The next variable (say  $X_p$ ) is included, which has the highest correlation with  $Y$  after removing  $X_m$
4. Build a model  $Y \sim X_m + X_p$ . If  $X_p$  is significant include it in the model else discard
5. Repeat steps (3) and (4) until reaching the stopping rule or running out of variables

# Forward selection method

Start with a NULL MODEL  
(a model with no predictors)

$$Y \sim$$

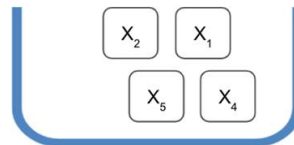
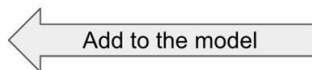
Consider 5 predictors



Obtain the most significant predictor  
(predictor having highest correlation with  $Y$ )

Model with most significant variable  
(say  $X_3$ )

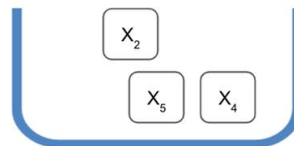
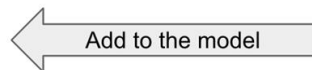
$$Y \sim \beta_0 + \beta_1 X_3$$



Obtain the next most significant predictor  
(from the remaining 4 predictor)

Model with most significant variable  
(say  $X_1$ )

$$Y \sim \beta_0 + \beta_1 X_3 + \beta_2 X_1$$



Continue until reaching the stopping  
rule or running out of variables



# Backward elimination method

## Procedure

1. Start with a full model (model with all  $k$  predictors)
2. Remove the variable which is least significant (variable with largest  $p$ -value)
3. Fit a new model with remaining  $(k-1)$  regressors
4. The next variable (say  $X_p$ ) is removed if it is least significant
5. Repeat steps (3) and (4) until reaching the stopping rule or all variables are significant

# Backward elimination method

Start with a FULL MODEL  
(a model with all the 5 predictors)

$$Y \sim \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5$$

Obtain the least significant predictor  
(predictor having highest p-value)

Model after removing the least significant variable  
(say  $X_3$  the least significant)

$$Y \sim \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_4 X_4 + \beta_5 X_5$$

Remove  $X_3$

Obtain the next least significant predictor  
(predictor having highest p-value after removing  $X_3$ )

Model after removing the least significant variable  
(say  $X_1$  is least significant)

$$Y \sim \beta_0 + \beta_2 X_2 + \beta_4 X_4 + \beta_5 X_5$$

Remove  $X_1$

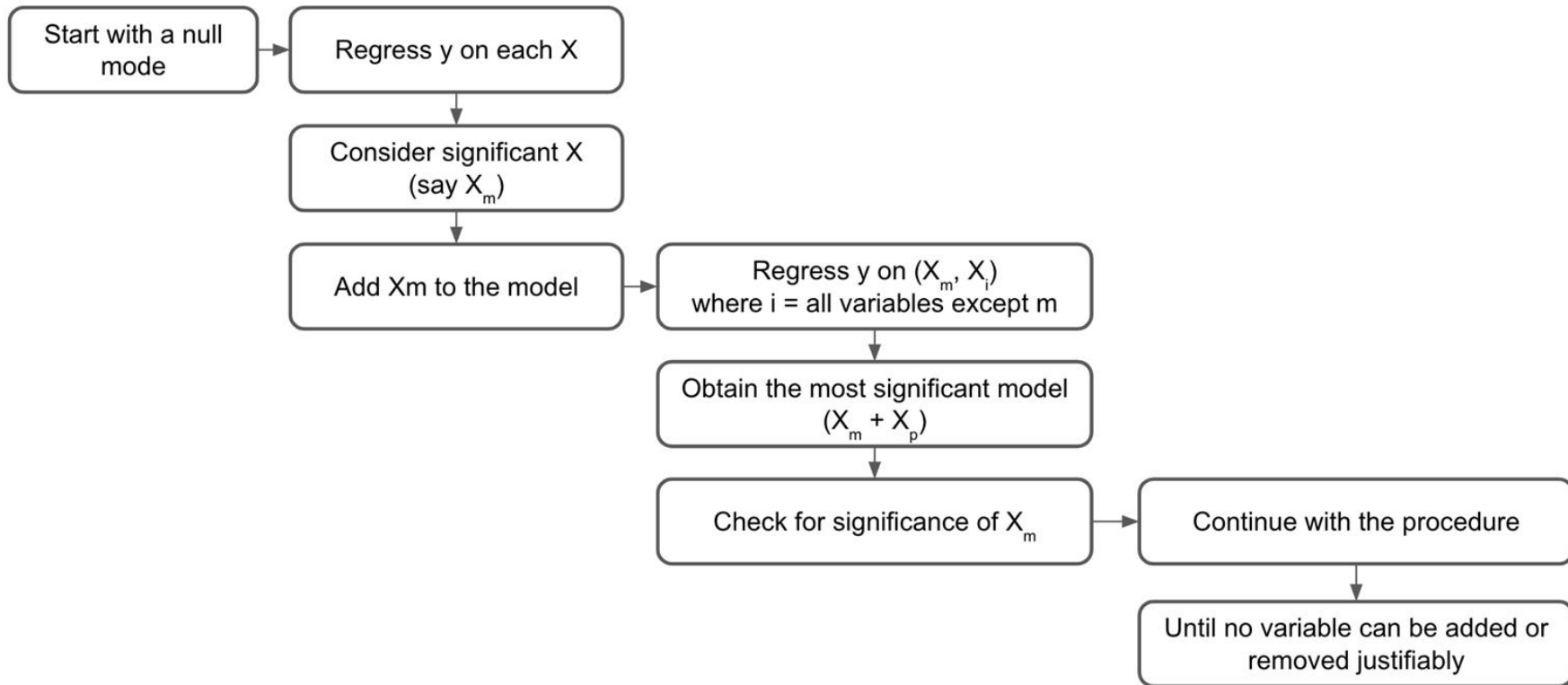
Continue until reaching the stopping  
rule or running out of variables



# Stepwise regression

- It is a combination of forward selection and backward elimination method
- Procedure:
  - Start with a null model (with no predictors)
  - At each step add or remove variable based on its corresponding p-value
  - Stop when no variable can be added or removed justifiably

# Stepwise regression



# Recursive feature elimination (RFE)

- It is an instance of backward feature elimination
- Procedure:
  - Train a full model
  - Create subsets for features
  - Set the subset size
  - Compute the ranking criteria for each feature subset
  - Remove the feature subset that has the least ranking

## 5. Recursive Feature Elimination (RFE)

It is the process that returns the significant features in the dataset by recursively removing the less significant feature subsets.

```
# set of independent variables
# drop the target variable using 'drop()'
# 'axis = 1' drops the specified column
X = df_energy_cons.drop('Appliances', axis = 1)

# consider the dependent variable
y = df_energy_cons['Appliances']

# split data into train subset and test subset
# set 'random_state' to generate the same dataset each time you run the code
# 'test_size' returns the proportion of data to be included in the testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 10)

# initiate Linear regression model to use in feature selection
linreg_rfe = LinearRegression()

# build the RFE model
# pass the regression model to 'estimator'
# pass number of required features to 'n_features_to_select'
# if we do not pass the number of features, RFE considers half of the features
rfe_model = RFE(estimator=linreg_rfe, n_features_to_select = 12)

# fit the RFE model on the training dataset using fit()
rfe_model = rfe_model.fit(X_train, y_train)

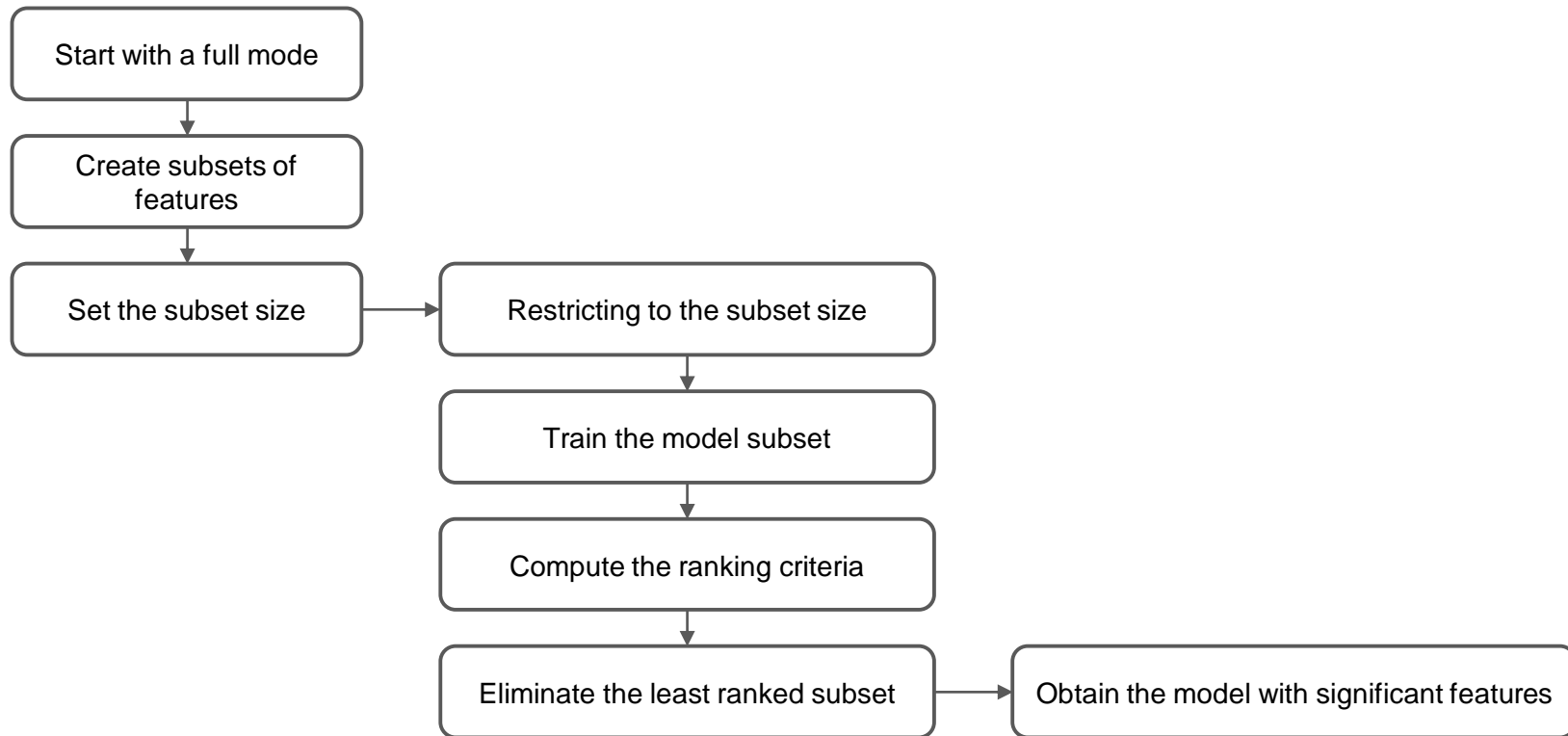
# create a series containing feature and its corresponding rank obtained from RFE
# 'ranking_' returns the rank of each variable after applying RFE
# pass the ranks as the 'data' of a series
# 'index' assigns feature names as index of a series
feat_index = pd.Series(data = rfe_model.ranking_, index = X_train.columns)

# select the features with rank = 1
# 'index' returns the indices of a series (i.e. features with rank=1)
signi_feat_rfe = feat_index[feat_index==1].index

# print the significant features obtained from RFE
print(signi_feat_rfe)

Index(['T1', 'RH_1', 'T2', 'RH_2', 'T3', 'T4', 'RH_4', 'T7', 'T8', 'RH_8',
      'T9', 'Windspeed'],
      dtype='object')
```

# Recursive feature elimination (RFE)



# Summary

Method	Properties
Forward Selection	Starts with a null model. Significant variables are added one at a time
Backward Elimination	Starts with a full model and removes the least significant feature at each step
Stepwise Method	Combination of forward selection and backward elimination method
Recursive Feature Elimination	Selects a best subset of features

# Optimization



# Optimization

## OPTIMIZATION

- Prediction Evaluation
- Model Validation
- Fine Tuning

- Prediction Evaluation: Process of evaluating how effectively the constructed model performs predictions
- Model Validation: Using test data to validate the model built using train data
- Fine Tuning: Maximizing the performance of a constructed model

# Prediction Evaluation

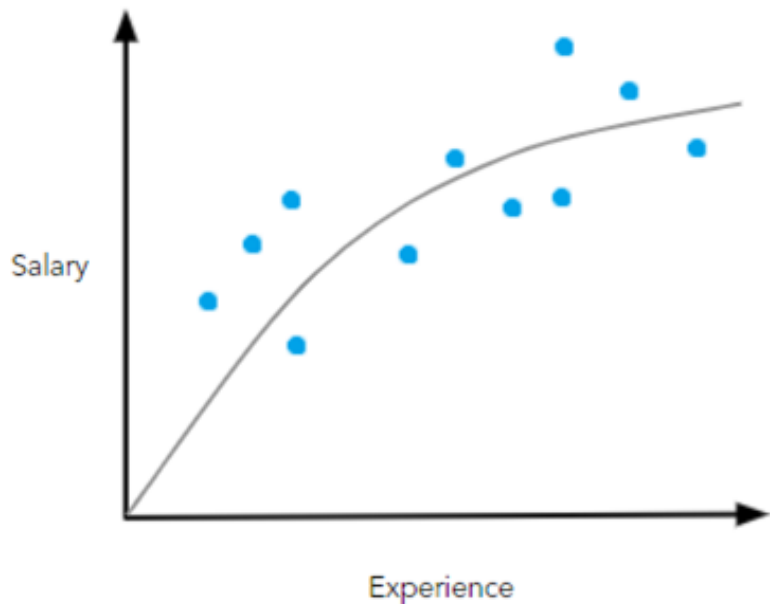
# Prediction Evaluation

To construct a model with high prediction efficacy it is important to consider the prediction errors:

- Bias
- Variance

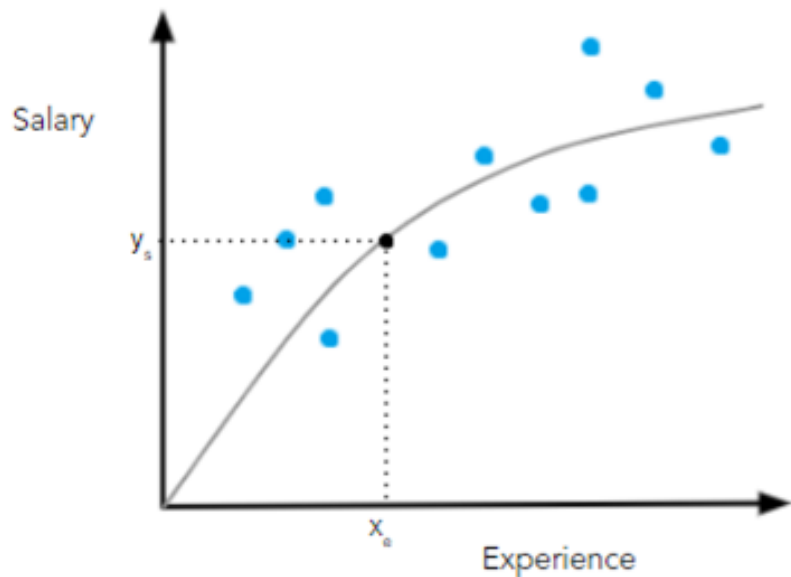
# Bias and Variance

# Bias and variance



- Consider the example of influence of years of experience on salary
- The plot represents a relationship between salary and experience
- Let us assume a grey curve that captures the true trend for the points in the plot

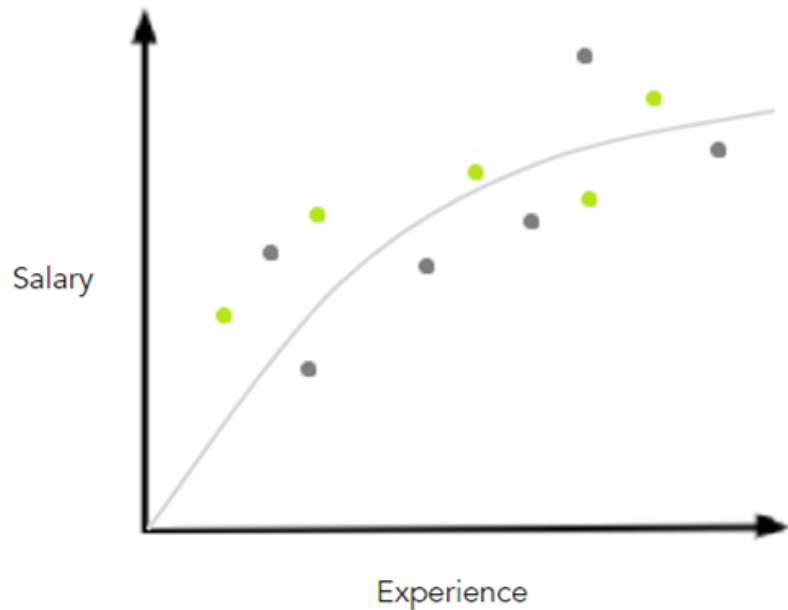
# Bias and variance



Given the years of experience information ( $x_e$ ), we can determine the salary ( $y_s$ ) using the grey curve

But we do not actually know the grey curve for this plot

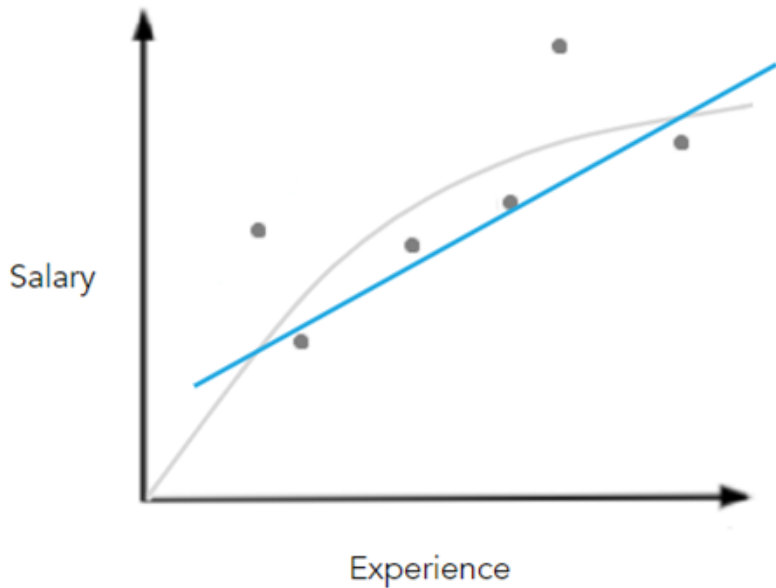
# Bias and variance



To find the curve that captures the true trend we divide the data into :

- Train data
- Test data

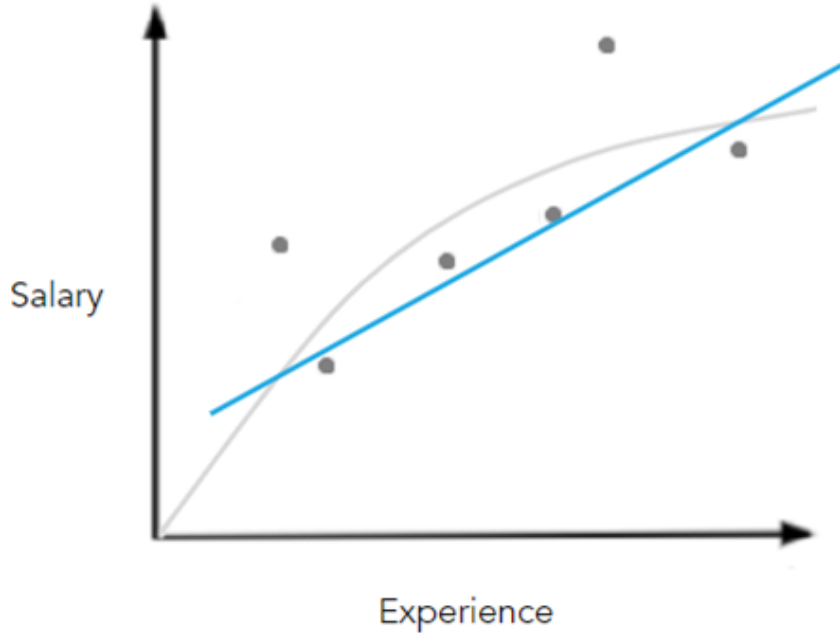
# Bias and variance



- We first estimate a regression line to capture the trend in the train data
- But compared to the line, the grey curve seems to better capture the relationship between experience and salary



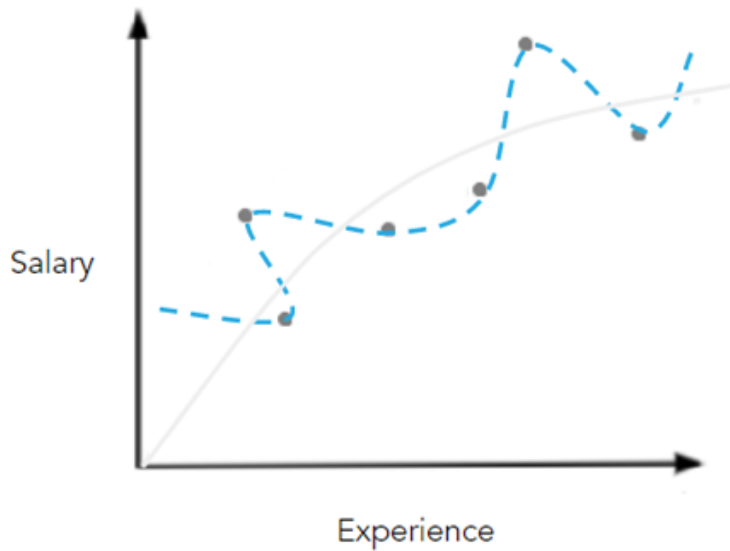
# Bias



- The linear regression line will never bend and hence will never capture the “true” relationship
- This inability to capture the “true” relationship is called **bias**

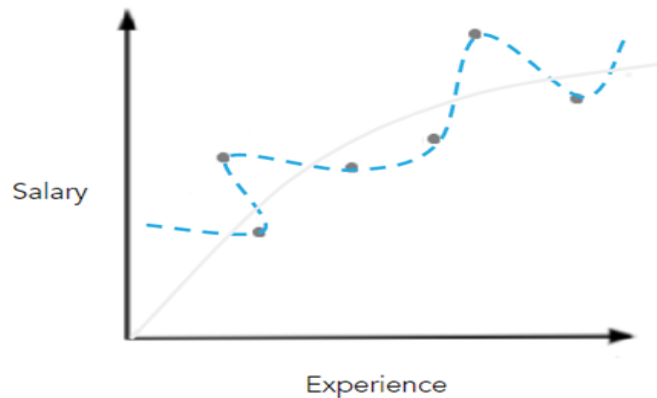
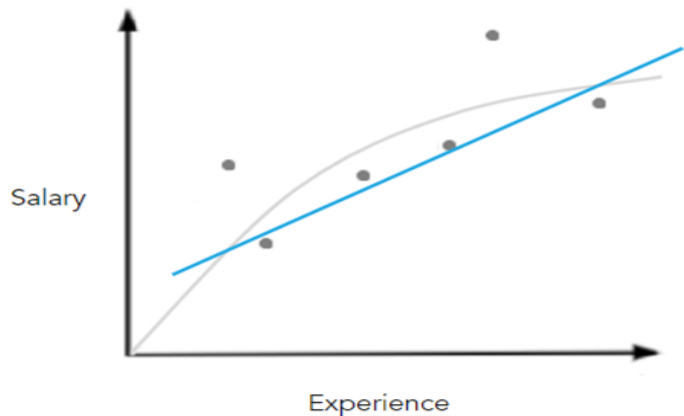
# Bias and variance

We then estimate a blue curve that captures the trend in train data perfectly, even better than the grey curve



# Error calculation

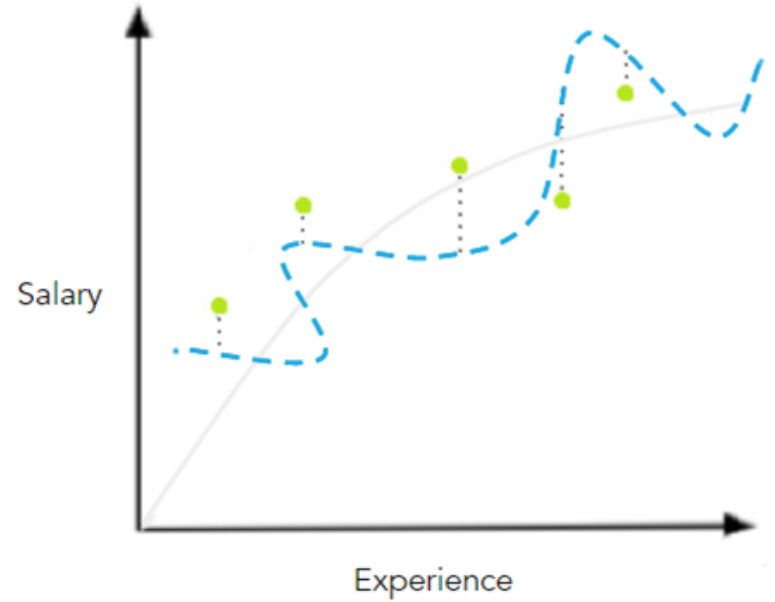
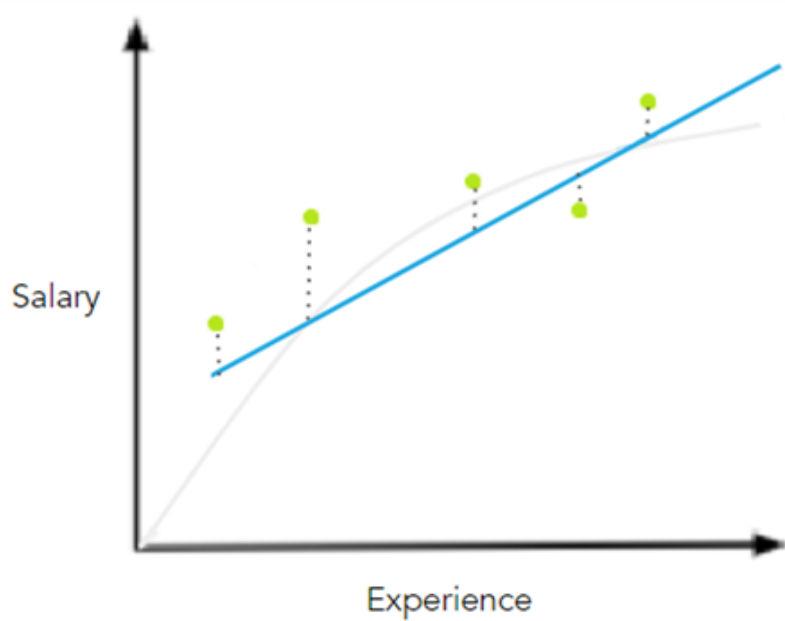
Error is measured by adding the squares of difference between the actual and the fitted values.



The curve fits the data points so perfectly, the difference between the actual and fitted values is actually zero.

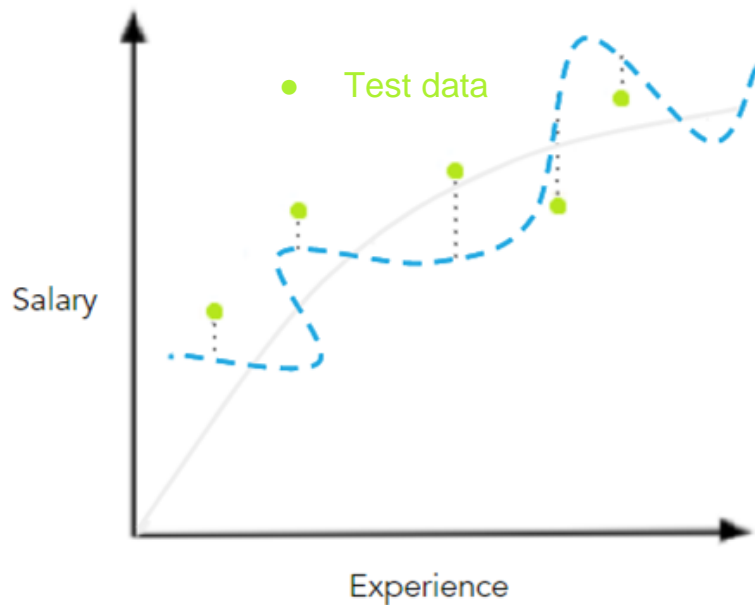
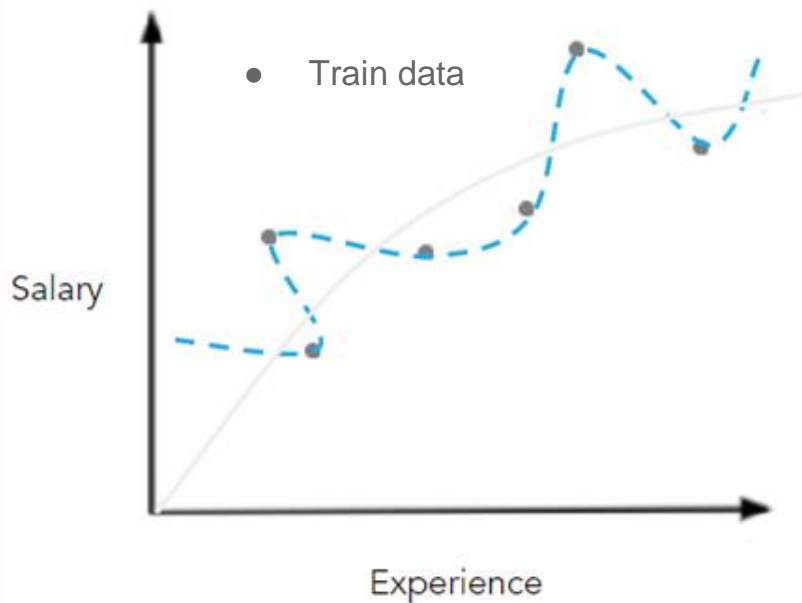
# Bias and variance

We use the same blue line and blue curve to estimate trends in test data



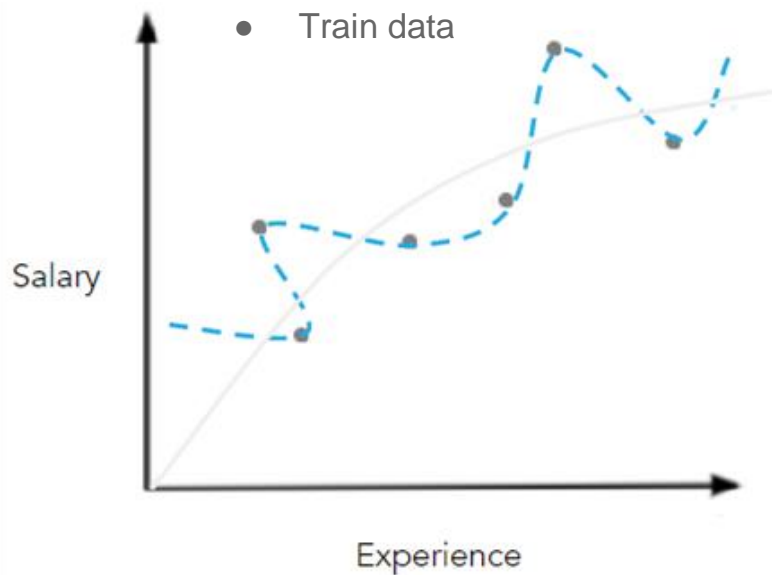
# Bias and variance

Even though the blue curve fits the train data with zero error, it does not predict well on test data



# Variance

This difference in fits is called **variance**



# Bias and variance

- Bias is the difference between a model's predicted values and the observed values
- Variance of a model is the difference between predictions if the model is fit to different datasets

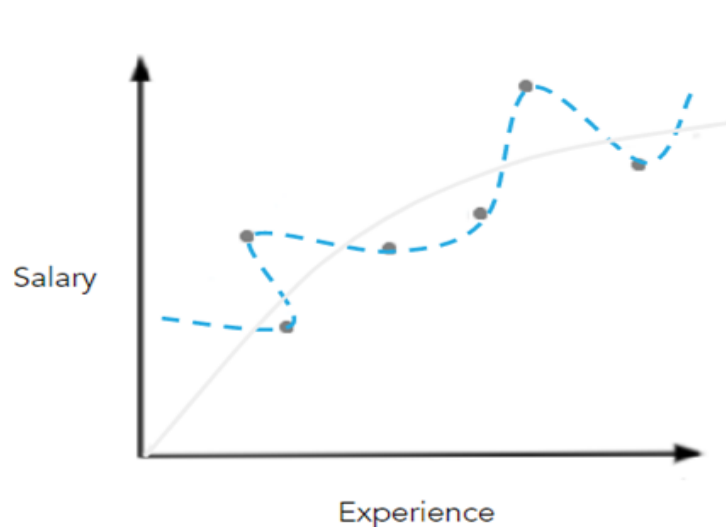
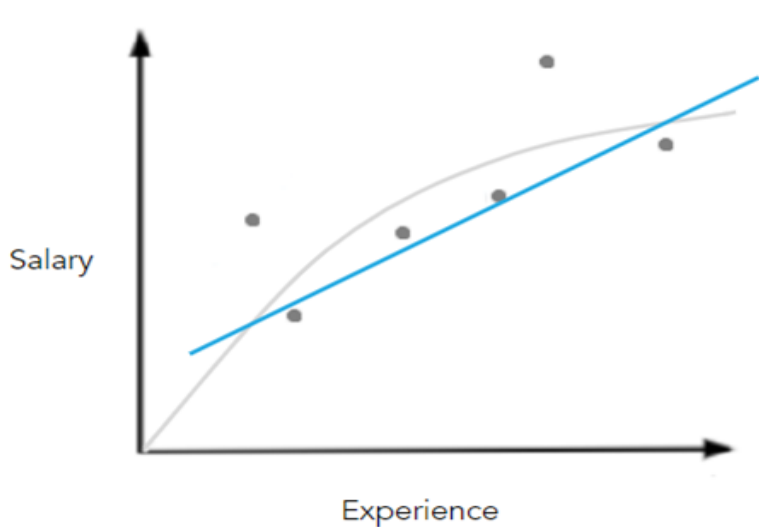
# Bias-Variance for a simple model

- If the model is too simple it will have high bias and low variance
- Such a model will give not perfectly accurate predictions, but the predictions will be consistent
- The model will not be flexible enough to learn from the majority of given data, this is termed as **underfitting**



# Example for bias

As we can see compared to the blue line, the blue curve captured the trend in train data perfectly. Hence we can say that the blue line has a high bias.



# Bias-Variance for a complex model

- If the model is too complex it will have low bias and high variance
- Such a model will give accurate predictions but inconsistently
- The high variance indicates it will have a much better fit on the train data compared to the test data, this is termed as **overfitting**

# Model Validation

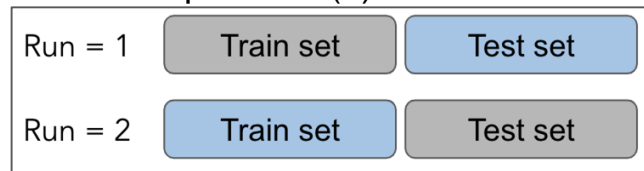
# Model validation

- The model validation methods use test data to validate the model built using train data
- The model validation:
  - k - fold cross validation
  - Leave one out cross validation (LOOCV)

# Cross validation

- Procedure:

- Consider a data having ' $2n$ ' observations
- Partition the dataset into two subsets: train and test sets of the equal size ( $n$ )
- Measure the model performance
- Swap the train and test sets
- Total error is obtained by summing up the errors for both runs



- This method is known as two fold cross validation
- Here, each observation is used exactly once for training and once for testing

# The k - fold cross validation

- Procedure:
  - Partition the dataset into 'k' subsets
  - Consider one subset as the test set and remaining subsets as train set
  - Measure the model performance
  - Repeat this until all k subsets are considered as test set
  - Total error is obtained by summing up the errors for all the k runs
- This method is known as the k - fold cross validation
- Here, each observation is used exactly k times for training and exactly once for testing



## Choosing k

The value of k is completely experimental and based on the size of the dataset. Usually it is considered as 5 or 10, but there is no hard rule for that.

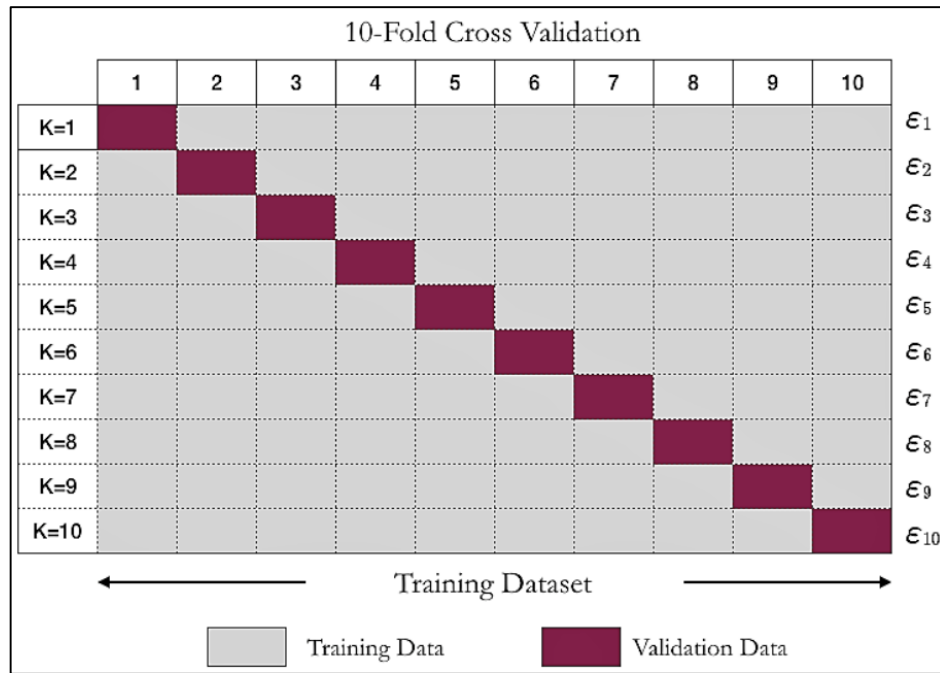
In python we can use the '[sklearn.model\\_selection.cross\\_val\\_score\(\)](#)' to perform the k-fold cross validation. By default, the function considers the value of  $k = 5$ .

# 10 - fold cross validation

Consider the 10-fold cross validation.

The total error is given by:

$$\epsilon = \frac{1}{10} \sum_{i=1}^{n=10} \epsilon_i$$





# LOOCV

- It is a special case of  $k$  - fold cross validation method. Instead of subsetting the data, at every run one observation is considered as the test set
- For  $n$  observations, there are  $n$  runs
- The total error is the sum of errors for  $n$  runs
- In LOOCV, the estimates from each fold are highly correlated and their average can have a high level of variance

Thank You