SuperDataScience

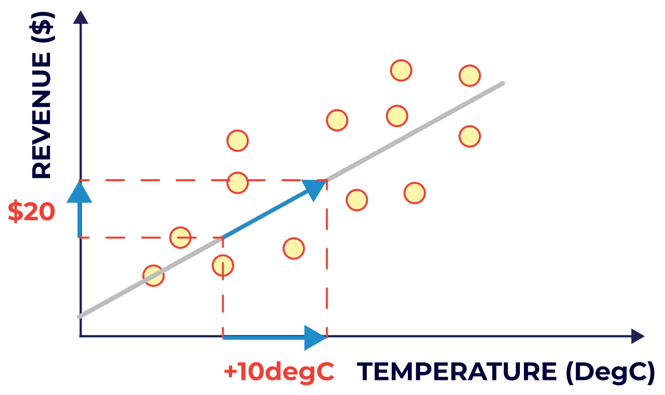
[2019]

MACHINE LEARNING REGRESSION MASTERCLASS

1. SIMPLE LINEAR REGRESSION

A. CONCEPT

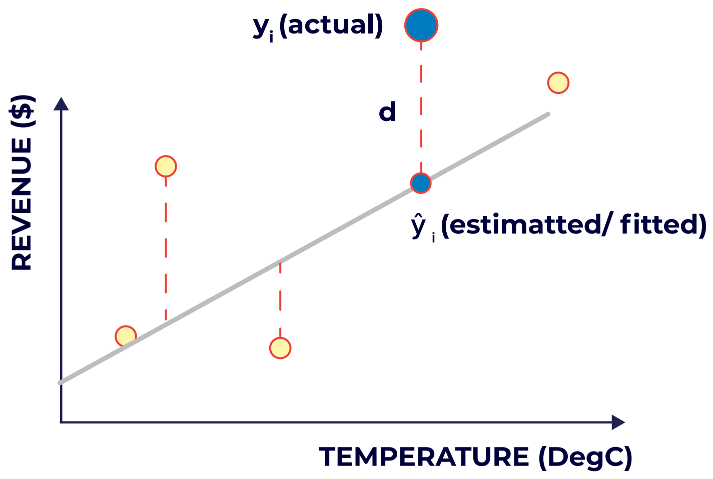
- **Simple linear regression is used to predict the value of one variable Y based on another variable X.**
- X is called the independent variable and Y is called the dependant variable.
- **Why simple?** Because it examines relationship between two variables only.
- **Why linear?** Because when the independent variable increases (or decreases), the dependent variable increases (or decreases) in a linear fashion.



$$\underbrace{y}_{\text{DEPENDENT VARIABLE REVENUE (\$)}} = \overbrace{\boxed{b} + \boxed{m}}^{\text{MODEL ! (GOAL)}} * \underbrace{x}_{\text{INDEPENDENT VARIABLE TEMPERATURE (DegC)}}$$

B. LEAST SQUARES CONCEPT

- Least squares fitting is a way to find the best fit curve or line for a set of points.
- The sum of the squares of the offsets (residuals) are used to estimate the best fit curve or line.
- Least squares method is used to obtain the coefficients m and b.



$$d = \hat{y}_i - y_i$$

$$\min \sum = \underbrace{(\hat{y}_i - y_i)}_{\text{MINIMUM (LEAST) SUM OF SQUARES}}$$

C. BUILD A SIMPLE LINEAR REGRESSION MODEL USING SCIKIT-LEARN

```
>> from sklearn.linear_model import LinearRegression
>> regressor = LinearRegression(fit_intercept = True)
>> regressor.fit(X_train,y_train)
>> print('Linear Model Coefficient (m): ', regressor.coef_)
>> print('Linear Model Coefficient (b): ', regressor.intercept_)
```

D. EVALUATE THE MODEL (MAKE PREDICTIONS USING TRAINED MODEL)

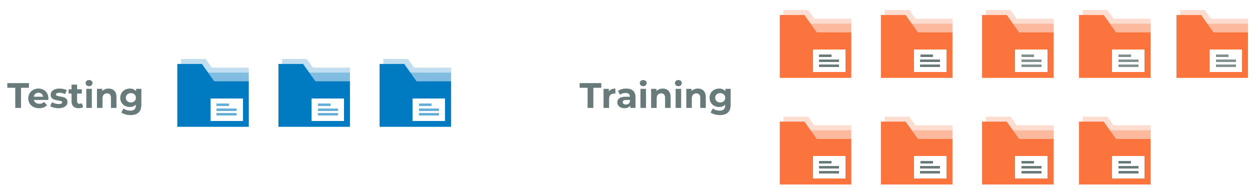
```
>> y_predict = regressor.predict( X_test)
>> y_predict
```

2. HOW TO DIVIDE DATASETS INTO TRAINING AND TESTING?

A. CONCEPT

Data set is generally divided into 75% for training and 25% for testing.

- **Training set:** used for model training.
- **Testing set:** used for testing trained model.



! **Make sure that testing dataset has never been seen by the trained model before.**

B. DIVIDING DATASET USING SCIKIT-LEARN

```
>> from import sklearn.model_selection train_test_split
>> X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.25, random_state=0)
```

3. HOW TO SCALE DATA USING SCIKIT-LEARN?

A. CONCEPT

Objective is to scale the training data to (0, 1) or (-1, 1). **This step is critical before training Artificial Neural Networks** to ensure that all inputs fed to the network are within similar range (i.e.: “treated fairly”). **If normalization is ignored, large inputs will dominate smaller ones.**

B. SCALING DATA USING SCIKIT LEARN

```
>> from sklearn.preprocessing import MinMaxScaler
>> scaler = MinMaxScaler()
>> X_scaled = scaler.fit_transform(X)
```

4. POLYNOMIAL REGRESSION

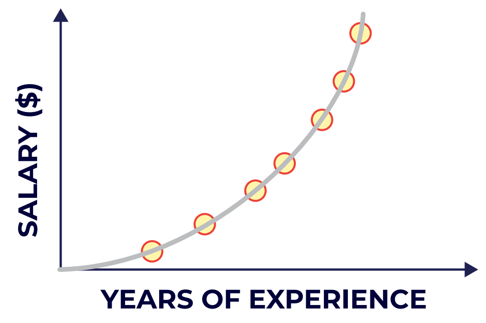
A. CONCEPT

Polynomial regression models is the relationship between the independent variable X and the dependent variable Y as an nth degree polynomial in X.

DEPENDENT VARIABLE

$$\boxed{y} = b_0 + b_1 * x + b_2 \boxed{x^2} + .. \, b_n x^n$$

INDEPENDENT VARIABLE



B. IMPLEMENT POLYNOMIAL REGRESSION IN SCIKIT-LEARN

```
>> from sklearn.preprocessing import PolynomialFeatures
>> poly_regressor = PolynomialFeatures(degree = 5)
>> X_columns = poly_regressor.fit_transform(X_train)
>> regressor = LinearRegression()
>> regressor.fit(X_columns, y_train)
>> print('Model Coeff:', regressor.coef_)
```

5. MULTIPLE LINEAR REGRESSION

A. CONCEPT

- **Multiple Linear Regression examines the relationship between more than two variables.**
- Recall that Simple Linear regression is a statistical model that examines linear relationship between two variables only.
- **Each independent variable has its own corresponding coefficient.**

DEPENDENT VARIABLES ←

$$\boxed{y} = b_0 + b_1 * \boxed{x_1} + b_2 \boxed{x_2} + .. \; b_n \boxed{x_n}$$

INDEPENDENT VARIABLES

B. IMPLEMENT MULTIPLE LINEAR REGRESSION IN SCIKIT-LEARN

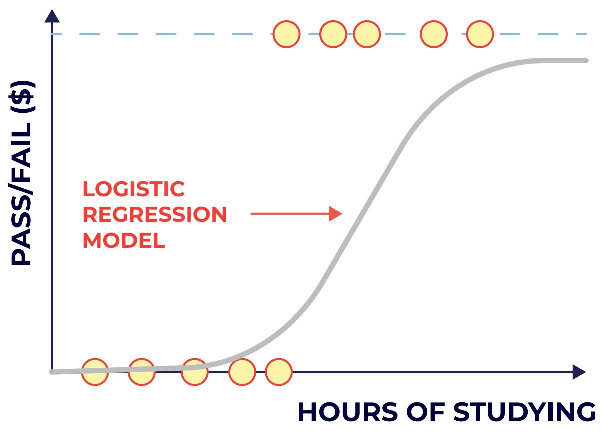
(Note: Similar to Simple Linear Regression)

```
>> from sklearn.linear_model import LinearRegression
>> regressor = LinearRegression(fit_intercept =True)
>> regressor.fit(X_train,y_train)
>> print('Linear Model Coefficient (m): ', regressor.coef_)
>> print('Linear Model Coefficient (b): ', regressor.intercept_)
```

6. LOGISTIC REGRESSION

A. CONCEPT

- **Logistic regression algorithm works by implementing a linear equation first with independent predictors to predict a value.**
- This value is then converted into a probability that could range from 0 to 1
- Logistic regression can be used as a classification technique by setting a threshold value (Ex: 0.5) to predict binary outputs with two possible values labeled "0" or "1"
- Therefore, **Logistic model output can be one of two classes: pass/fail, win/lose, healthy/sick**



Linear equating $y=b_0+b_1*x$

Apply Sigmoid function

$$P(x)=\text{sigmoid}(y)$$

$$P(x)=\frac{1}{1+e^{-y}}$$

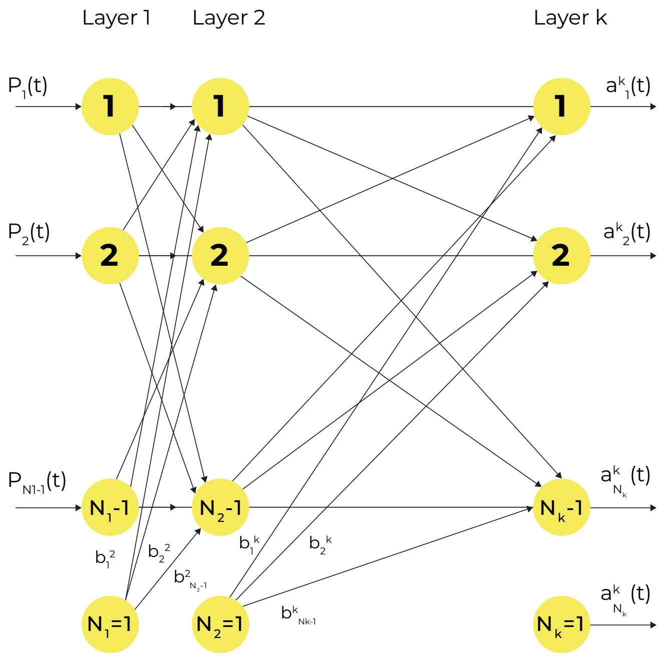
$$P(x)=\frac{1}{1+e^{-(b_0+b_1*x)}}$$

B. LOGISTIC REGRESSION IN SCIKIT LEARN

```
>> from sklearn.linear_model import LogisticRegression
>> Logistic_Regressor = LogisticRegression(random_state = 0)
>> Logistic_Regressor.fit(X_train, y_train)
```

7. ARTIFICIAL NEURAL NETWORKS (ANNs) TO PERFORM REGRESSION

A. CONCEPT



Artificial Neural Networks are information processing models inspired by the human brain. ANNs are built in a layered fashion where inputs are propagated starting from the input layer through the hidden layers and finally to the output.

Networks training is performed by optimizing the matrix of weights outlined below:

$$\begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1,N} \\ W_{21} & W_{22} & \cdots & W_{2,N} \\ \vdots & & \ddots & \\ W_{m-1,1} & W_{m-1,2} & \cdots & W_{m-1,N_1} \\ W_{m,1} & W_{m,2} & \cdots & W_{m,N_1} \end{bmatrix}$$

B. BUILD AN ANNs USING KERAS

```
>> import tensorflow.keras
>> from keras.models import Sequential
>> from keras.layers import Dense
>> from sklearn.preprocessing import MinMaxScaler

>> model = Sequential()
>> model.add(Dense(25, input_dim=5, activation='relu'))
>> model.add(Dense(25, activation='relu'))
>> model.add(Dense(1, activation='linear'))
>> model.summary()
```

C. TRAIN AN ANN USING KERAS

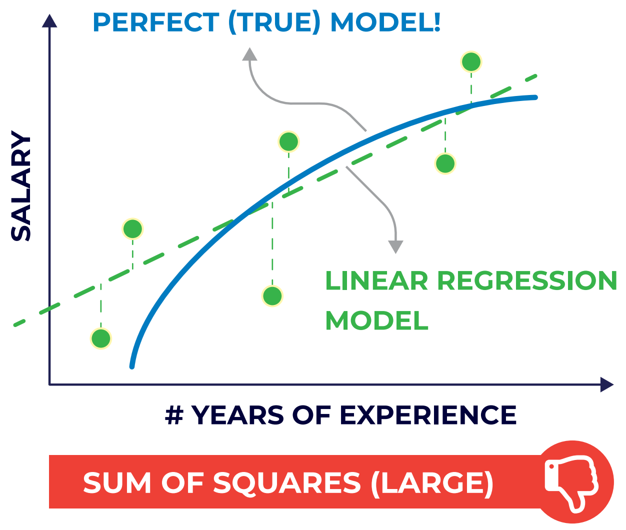
```
>> model.compile(optimizer='adam', loss='mean_squared_error')
>> epochs_hist = model.fit(X_train, y_train, epochs=20, batch_size=25, validation_split=0.2)
```

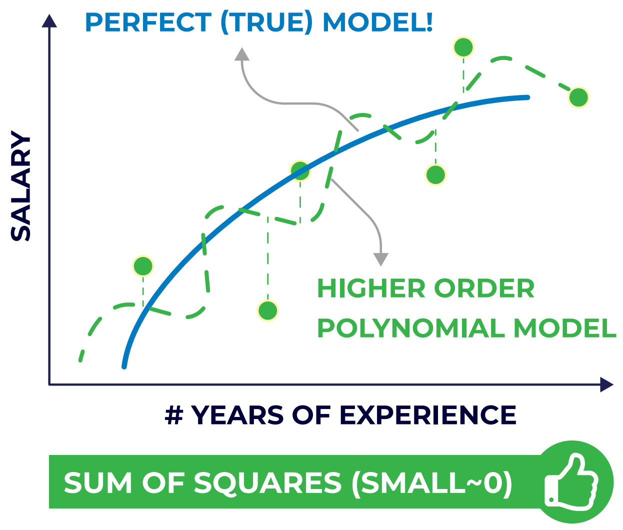
D. EVALUATE THE TRAINED ANN MODEL

```
>> X_Testing = np.array([[input #1, input #2, input #3,..., input #n]])
>> y_predict = model.predict(X_Testing)
```

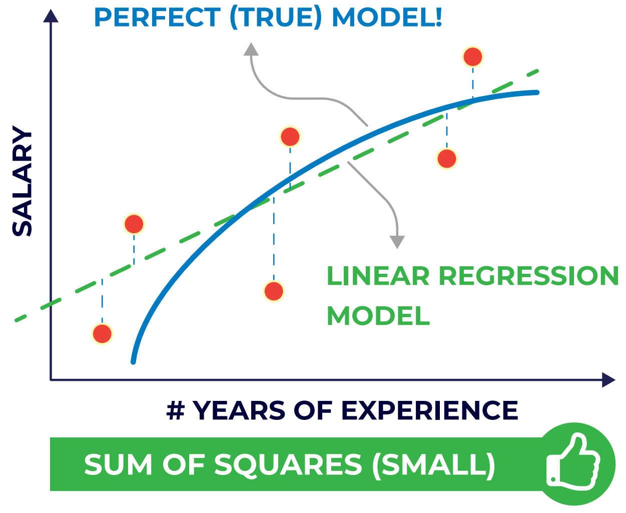
8. BIAS-VARIANCE TRADEOFF

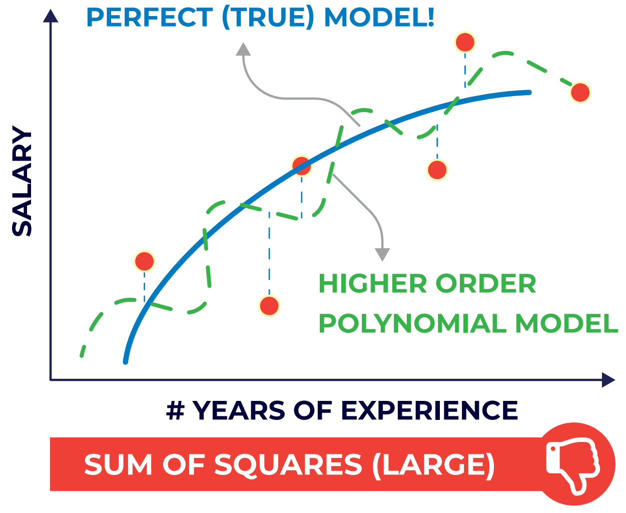
BIAS AND VARIANCE:
Model #1 vs Model #2 during Training





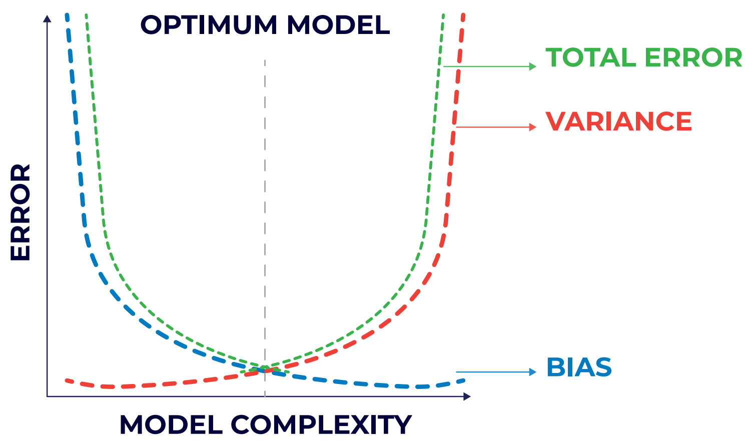
BIAS AND VARIANCE: Model #1 vs Model #2 during Testing





The polynomial model performs poorly on the testing dataset and therefore it has large variance

MODEL #1 (LINEAR REGRESSION) (SIMPLE)	MODEL #2 (HIGH ORDER POLYNOMIAL) (COMPLEX)
Model #1 has High bias because it is very rigid (not flexible) and cannot fit the training dataset well.	**Model #2 has Small bias** because it is flexible and can fit the training dataset very well.
Model #1 has Small variance (variability) because it can fit the training data and the testing data with similar level (the model is able to generalize better) and avoids overfitting.	**Model #2 has Large variance (variability)** because the model over fitted the training dataset and it performed poorly on the testing dataset.
Model #1 Performance is consistent between the training dataset and the testing dataset.	**Model #2 performance varies greatly between the training dataset and the testing dataset (high variability).**
Good generalization.	**Over fitted.**



9. RIDGE AND LASSO REGRESSION

A. RIDGE REGRESSION

- **Ridge regression works by increasing the bias to improve variance** (model generalization capability)
- **Ridge regression works by changing the slope of the regression model line** by adding the following penalty:

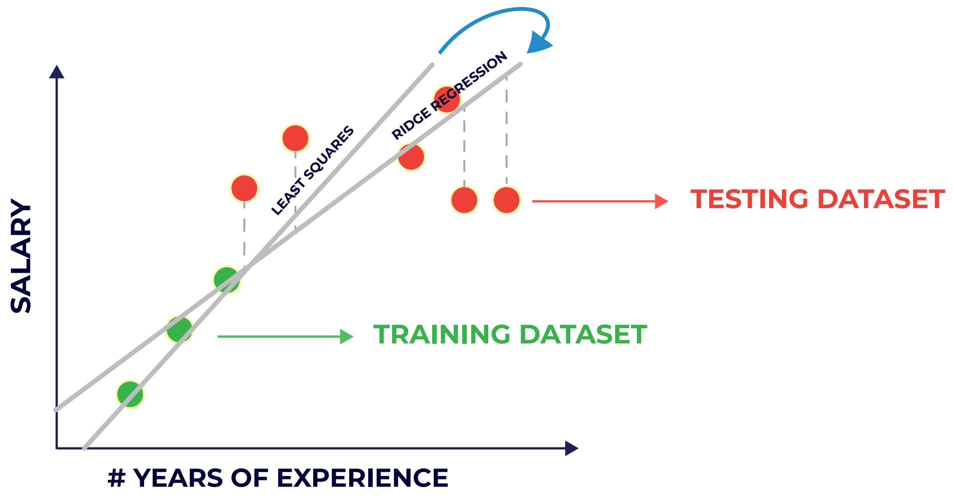
$$\alpha * \text{slope}^2$$

- The model performance might be little poor on the training set but it will perform consistently well on both the training and testing datasets.
- **Least Squares Regression:**

$$\text{Min (sum of squared residuals)}$$

- **Ridge Regression:**

$$\text{Min (sum of squared residuals} + \alpha * \text{slope}^2 \text{)}$$



B. LASSO REGRESSION

- **Lasso Regression is similar to Ridge regression**
- It works by introducing a bias term but instead of squaring the slope, the absolute value of the slope is added as a penalty term
- **Lasso Regression:**

$$\text{Min (sum of squared residuals} + \alpha * |\text{slope}| \text{)}$$

- The effect of Alpha on Lasso regression is similar to its effect n ridge regression As Alpha increases, the slope of the regression line is reduced and becomes more horizontal.

10. REGRESSION METRICS

A. MEAN ABSOLUTE ERROR (MAE)

- **Mean Absolute Error (MAE) is obtained by calculating the absolute difference** between the model predictions and the true (actual) values.
- **MAE is a measure of the average magnitude** of error generated by the regression model.
- **The mean absolute error (MAE) is calculated as follows:**

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- **MAE is calculated by following these steps:**

$$\begin{tikzcd} \boxed{\text{Calculate the residual for every data point}} \arrow[r] & \boxed{\text{Calculate the absolute value (to get rid of the sign)}} \arrow[r] & \boxed{\text{Calculate the average of all residuals}} \end{tikzcd}$$

- **If MAE is zero, this indicates that the model predictions are perfect.**

B. MEAN SQUARE ERROR (MSE)

- **Mean Square Error (MSE) is very similar to the Mean Absolute Error (MAE)** but instead of using absolute values, squares of the difference between the model predictions and the training dataset (true values) is being calculated.
- MSE values are generally large compared to the MAE since the residuals are being squared.
- In case of data outliers, MSE will become much larger compared to MAE
- In MSE, error increases in a quadratic fashion while the error increases in proportional fashion in MAE
- **The MSE is calculated as follows:**

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

- **MAE is calculated by following these steps:**

$$\begin{tikzcd} \boxed{\text{Calculate the residual for every data point}} \arrow[r] & \boxed{\text{Calculate the squared value of the residuals}} \arrow[r] & \boxed{\text{Calculate the average of all residuals}} \end{tikzcd}$$

C. MEAN ABSOLUTE ERROR (MAE)

- **Root Mean Square Error (RMSE) represents the standard deviation of the residuals** (i.e.: differences between the model predictions and the true values (training data)).
- RMSE can be easily interpreted compared to MSE because RMSE units match the units of the output.
- RMSE provides an estimate of how large the residuals are being dispersed.
- **The MSE is calculated as follows:**

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

- **RMSE is calculated by following these steps:**

Calculate the residual for every data point → **Calculate the squared value of the residuals** → **Calculate the average of the squared residuals** → **Obtain the square root of the result**

D. MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)

- **MAE values can range from 0** to infinity which makes it difficult to interpret the result as compared to the training data.
- **Mean Absolute Percentage Error (MAPE) is the equivalent to MAE** but provides the error in a percentage form and therefore overcomes MAE limitations.
- MAPE might exhibit some limitations if the data point value is zero (since there is division operation involved)
- **The MAPE is calculated as follows:**

$$\text{MAPE}=\frac{100\%}{n}\sum_{i=1}^{n}|(y_i-\hat{y}_i)/y_i|$$

E. MEAN PERCENTAGE ERROR (MPE)

- **MPE is similar to MAPE** but without the absolute operation.
- MPE is useful to provide an insight of how many positive errors as compared to negative ones.
- **The MPE is calculated as follows:**

$$\text{MAPE}=\frac{100\%}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)/y_i$$