Logistic Regression

This is my ATM Project (Amati (observe), Tiru (imitate), Modifikasi (modification))
Learn into ML DL AI as the beginner to be a hero!

Import Libraries

In [1]:
```
import pandas as pd # data manipulation
import numpy as np # array manipulation
%matplotlib inline
import matplotlib.pyplot as plt # visualization
import sklearn
from sklearn.compose import ColumnTransformer # transform the column
from sklearn.preprocessing import OneHotEncoder # encode datatype
from sklearn.linear_model import LinearRegression # modelling
from sklearn.model_selection import train_test_split # splitting train and test
from sklearn.metrics import mean_squared_error # evaluation
```

Import Dataset

In [2]:
```
dt = pd.read_csv("data_input/insurance.csv")
```

In [3]:
```
dt.head()
```

Out[3]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [4]:
```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

In this step, we importing the dataset from data_input folder. This data has 1071 rows with 7 columns. The data type is still not relevan, so we turn it into the right types.

Data Wrangling

In this step, we would check if there is any missing values, unappropriate data types, duplicates, etc.

In [18]:
```
dt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1337 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1337 non-null   int64  
 1   sex       1337 non-null   object 
 2   bmi       1337 non-null   float64
 3   children  1337 non-null   int64  
 4   smoker    1337 non-null   object 
 5   region    1337 non-null   object 
 6   charges   1337 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 83.6+ KB
```

In [ ]:
```
dt[['sex','smoker','region']] = dt[['sex','smoker','region']].astype('category')
```

Missing value

In [5]:
```
dt.isnull().any()
```

Out[5]:
```
age         False
sex         False
bmi         False
children    False
smoker      False
region      False
charges     False
dtype: bool
```

In [10]:
```
dt.duplicated().any()
```

Out[10]:
```
True
```

In [12]:
```
dt.drop_duplicates(inplace = True)
```

In [13]:
```
dt.duplicated().any()
```

Out[13]:
```
False
```

Splitting Data

Split data into target and its predictors.

In [15]:
```
# get the target
y = dt['charges']

# drop target to get predictors
X = dt[['age','children']]
```

In [16]:
```
# split 80% train and 20% test
X_train, X_test, y_train, y_test = train_test_split(X, \
                                                    y,\
                                                    test_size=0.2,\
                                                    random_state=42)
```

Training

Fitting data using train that has been splitted in previous step. We can use LinearRegression function from sklearn.

In [17]:
```
mdl = LinearRegression()
mdl.fit(X_train, y_train)
```

Out[17]:
```
LinearRegression()
```

Testing

Predict model that has builded with test data

In [22]:
```
y_pred = mdl.predict(X_test)
y_pred[:5]
```

Out[22]:
```
array([14630.08887853, 12588.95412289, 17160.86493902, 12382.48008346,
       11698.16755848])
```

Model Evaluation

In [24]:
```
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error : ", mse)
```

```
Mean Squared Error :  166847445.41074288
```

Our model has 16% error which can tell that our model is quite good for predicting new data. Next, we predict new (dummy) data

In [28]:
```
new_dt = {'age' : [50,30,20],
          'children' : [7, 0, 4]}

new_dt = pd.DataFrame(new_dt)
```

In [29]:
```
y_hat = mdl.predict(new_dt)
y_hat
```

Out[29]:
```
array([19659.19579616, 10090.62326575, 10438.68093789])
```

Our model success to predict new data. If you have any advice for this work or suggestion what i should learn next, you can send an email (mailto:rahfairuzran@gmail.com) to me