Multiple Linear Regression

Cross Validation - I

Contents

Cross Validation in Predictive Modeling

Cross Validation is a

process of evaluating the model on

'Out of Sample' data

- **Model performance measures** such as R-squared or Root Mean Squared Error (RMSE) **tend to be optimistic on 'In sample data'**

- **Model performance on out of sample data** gives more realistic picture of model performance.

Cross validation is important because although a model is built on historical data, ultimately it is to be used on future data. However good the model, if it fails on out of sample data then it defeats the purpose of predictive modeling.

Cross Validation in Predictive Modeling

There are different approaches to cross validation. Five most important of them are:

- Hold-Out Validation
- K-Fold Cross Validation
- Repeated K-Fold Cross Validation
- Leave-One-Out Cross Validation (LOOCV)
- Re-sampling Validation Method (Bootstrap Method)

Case Study – Modeling Motor Insurance Claims

Background

- A car insurance company collects range of information from their customers at the time of buying and claiming insurance. The company wishes to check if any of this information can be used to model and predict claim amount

Objective

- To model motor insurance claim amounts based on vehicle related information collected at the time of registering and claiming insurance

Available Information

- Sample size is 1000
- Independent Variables: Vehicle Information – **Vehicle Age, Engine Capacity, Length and Weight of the Vehicle**
- Dependent Variable: **Claim Amount**

Data Snapshot

Motor_Claims

Independent variables

Dependent variable

Observations

vehage	CC	Length	Weight	claimamt
4	1495	4250	1023	72000
2	1061	3495	875	72000
2	1405	3675	980	50400
7	1298	4090	930	39960
2	1495	4250	1023	106800
1	1086	3565	854	69592.8

Columns	Description	Type	Measurement	Possible values
vehage	Age of the vehicle at the time of claim	integer	Years	positive values
CC	Engine capacity	numeric	cc	positive values
Length	Length of the vehicle	numeric	mm	positive values
Weight	Weight of the vehicle	numeric	kg	positive values
claimamt	Claim amount	numeric	INR	positive values

Data Visualization

```
#Importing the Data
```

```
import pandas as pd
motor=pd.read_csv('Motor_Claims.csv')
```

```
# Install package “seaborn” if not installed previously
# Obtain  scatter plot matrix
```

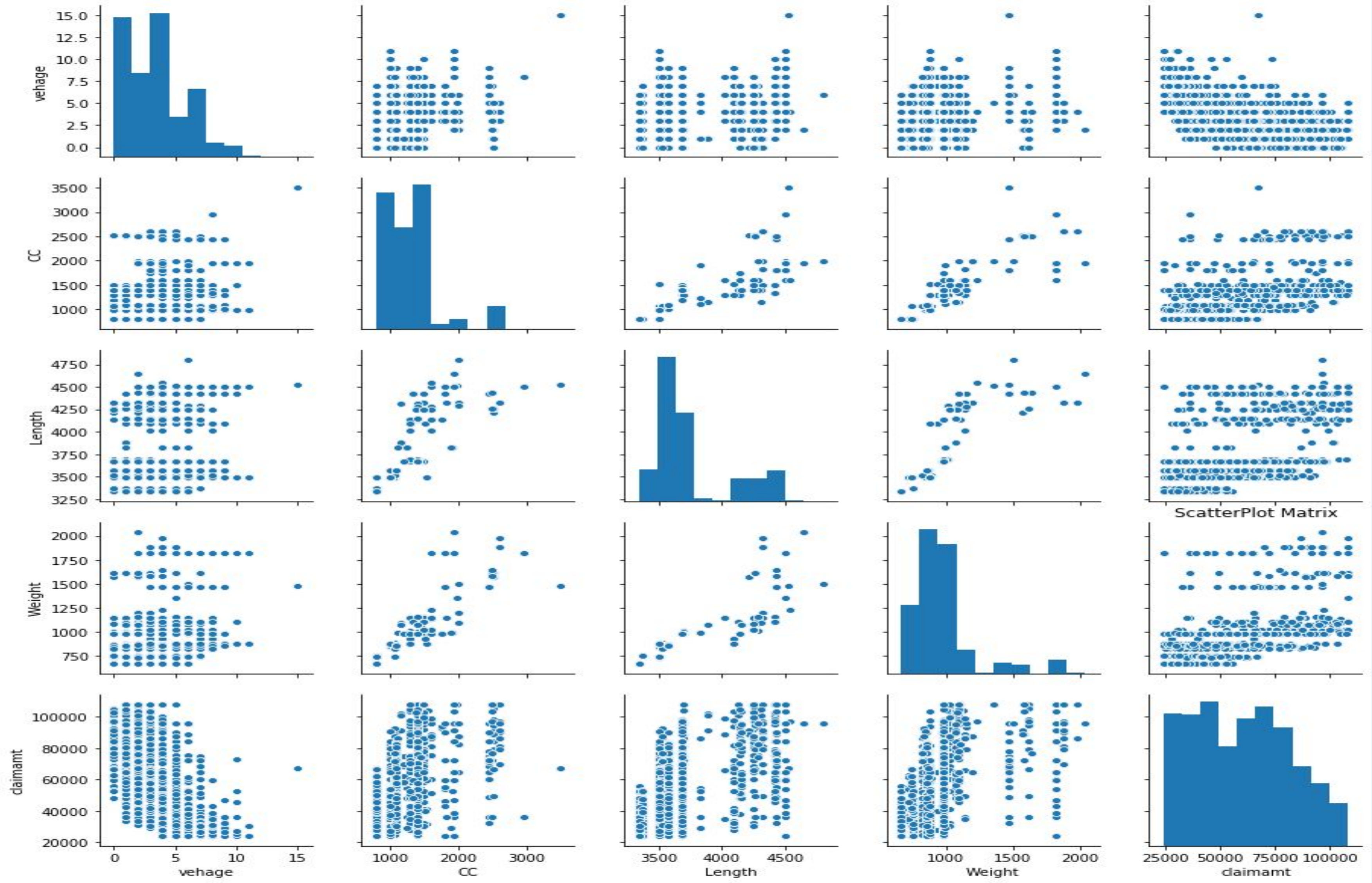
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.pairplot(motor);plt.title('ScatterPlot Matrix')
```

Using the pairplot function in the seaborn library to get a scatter plot of the variables in the data set

Scatter Plot

```
# Output
```



Interpretation :
The scatter plot matrix gives an indication of multicollinearity.

Modeling Using ols Function

```
# Linear regression model
import statsmodels.formula.api as smf
motormodel = smf.ols('claimamt~Length+CC+vehage+Weight', data=motor).fit()
motormodel.summary()
```

```
# Output
                            OLS Regression Results
==============================================================================
Dep. Variable:               claimamt   R-squared:                       0.738
Model:                            OLS   Adj. R-squared:                  0.737
Method:                 Least Squares   F-statistic:                     700.3
Date:                Fri, 25 Oct 2019   Prob (F-statistic):          1.83e-287
Time:                        16:38:15   Log-Likelihood:                -10754.
No. Observations:                1000   AIC:                         2.152e+04
Df Residuals:                     995   BIC:                         2.154e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -5.477e+04   5569.375     -9.833      0.000   -6.57e+04   -4.38e+04
Length        35.4607      1.990     17.824      0.000      31.557      39.365
CC            15.4133      2.114      7.292      0.000      11.265      19.561
vehage     -6637.2134    154.098    -43.071      0.000   -6939.607   -6334.820
Weight       -16.2547      3.678     -4.420      0.000     -23.472      -9.038
==============================================================================
Omnibus:                        7.335   Durbin-Watson:                   2.094
Prob(Omnibus):                  0.026   Jarque-Bera (JB):                9.587
Skew:                          -0.058   Prob(JB):                      0.00828
Kurtosis:                       3.466   Cond. No.                     6.33e+04
==============================================================================
```

Interpretation:
All independent variables in the model are significant.

Detecting Multicollinearity

```
# Obtaining vif
from patsy import dmatrices
from statsmodels.stats.outliers_influence import
variance_inflation_factor

# Break into left and right hand side; y and X
y, X = dmatrices('claimamt~Length+CC+vehage+Weight', data=motor,
return_type="dataframe")

#Calculating VIF
vif = pd.Series([variance_inflation_factor(X.values, i)for i in
range(X.shape[1])],index=X.columns)
vif
```

```
Intercept    240.261728
Length         3.396171
CC             5.881428
vehage         1.038357
Weight         6.552811
dtype: float64
```

- *variance_inflation_factor in library statsmodels gives the VIFs of the independent variables in the regression model.*

Interpretation:

- *CC and Weight have VIF >5*

Re- Modeling

```
# New model
motormodel1 = smf.ols('claimamt~Length+CC+vehage', data=motor).fit()
motormodel1.summary()
```

New model after removing weight to remove multicollinearity.

```
# Output of the new model
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:               claimamt   R-squared:                       0.733
Model:                            OLS   Adj. R-squared:                  0.732
Method:                 Least Squares   F-statistic:                     910.3
Date:                Thu, 31 Oct 2019   Prob (F-statistic):          8.79e-285
Time:                        12:48:57   Log-Likelihood:                -10764.
No. Observations:                1000   AIC:                         2.154e+04
Df Residuals:                     996   BIC:                         2.156e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   -4.92e+04   5475.151     -8.985      0.000   -5.99e+04   -3.85e+04
Length        32.0652      1.852     17.312      0.000      28.431      35.700
CC             8.6886      1.481      5.867      0.000       5.783      11.595
vehage     -6638.0765    155.525    -42.682      0.000   -6943.270   -6332.883
==============================================================================
Omnibus:                       10.930   Durbin-Watson:                   2.081
Prob(Omnibus):                  0.004   Jarque-Bera (JB):               15.892
Skew:                          -0.072   Prob(JB):                     0.000354
Kurtosis:                       3.600   Cond. No.                     5.99e+04
==============================================================================
```

Interpretation:
All independent variables in the model are significant.

* Dropping one independent variable is one of the remedial measures to adjust for multicollinearity (when not many variables are multicollinear). As weight had the maximum VIF value, it is excluded from the model to adjust for multicollinearity.

VIF of New Model

```
# VIF
```

```
# Break into left and right hand side; y and X
y, X = dmatrices('claimamt~Length+CC+vehage', data=motor,
return_type="dataframe")

#Calculating VIF
vif = pd.Series([variance_inflation_factor(X.values, i)for i in
range(X.shape[1])],index=X.columns)
vif
```

Getting VIFs of the independent variables in the new model

```
# VIFs of variables in the new model
```

```
Intercept    227.959103
Length         2.889718
CC             2.833931
vehage         1.038355
dtype: float64
```

Interpretation:
All VIF s are <5.

RMSE of the Model

```
# RMSE of the model
```

```
motor=motor.assign(res=pd.Series(motormodel1.resid))

from math import sqrt
RMSE = sqrt((motor['res']**2).mean())
RMSE
```

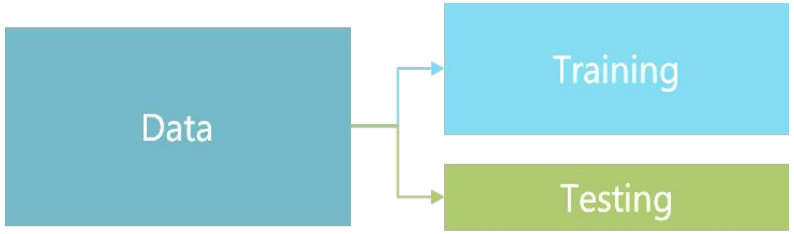
```
# Output
```

```
11444.512861029943
```

Interpretation :
RMSE for the model is 1144.51

* RMSE of this model will be used later to measure the performance of the model on cross validation

Hold-Out Validation



In Hold-Out validation method, **available data is split into two non-overlapped parts: 'Training Data' and 'Testing Data'**

- The model is,
  - Developed using training data
  - Evaluated using testing data

Training data should have more sample size. **Typically 70%-80% data is used for model development**

Hold Out Validation in Python

```
# Import train_test_split
# Creation of Datasets for Validation
import numpy as np
from sklearn.model_selection import train_test_split
import statsmodels.formula.api as smf

motor=pd.read_csv('Motor_Claims.csv')

motor_train, motor_test = train_test_split(motor, test_size=0.2,
random_state=0)
```

We use the function train_test_split to split the data 80 -20, with test_size=0.2 specifying that 20 % of data used as test data

```
motor_train.shape
motor_test.shape
```

training and testing data sets for the Motor Insurance Data

```
# Output
(800, 5)
(200, 5)
```

Dimension of training set

Dimension of testing set

Hold Out Validation in Python

```
# RMSE of training data
```

```
motor_model=smf.ols('claimamt~vehage+CC+Length', data =
motor_train).fit()

motor_train=motor_train.assign(res=pd.Series(motor_model.resid))
motor_train.head()

RMSEtrain=pd.Series(np.sqrt((motor_train.res)**2).mean())
RMSEtrain
```

```
# RMSE for testing data
```

```
motor_test=motor_test.assign(pred=pd.Series(motor_model.predict(motor_
test)))
motor_test=motor_test.assign(res=pd.Series(motor_test.claimamt -
motor_test.pred))

RMSEtest=pd.Series(np.sqrt((motor_test.res)**2).mean())
RMSEtest
```

*

For testing data, since we work with predictors, we calculate residuals as observed values minus the predicted values
Note : Since data is randomly taken, output will differ slightly

Hold Out Validation in Python

```
# Output
```

	vehage	CC	Length	Weight	claimamt	res
0	4	1495	4250	1023	72000.0	-3603.862083
2	2	1405	3675	980	50400.0	-17806.818626
3	7	1298	4090	930	39960.0	-8741.177051
4	2	1495	4250	1023	106800.0	18045.949781
6	4	796	3495	740	38400.0	-4537.892032

res column gives the residual for the training set data

```
11444.512861029943
```

RMSE for the original data with all data points

```
11456.386840541307
```

RMSE for the training data

```
10846.219628859553
```

RMSE for testing data

Interpretations :

- *Comparing RMSE of training and testing data shows not much difference between the two and also are in line with the RMSE of the original model. Thus we can say that the model is stable.*

* **There is no rule of thumb for comparison of model performance. The only criterion is that performance measure for training and testing data should not be too diverse**

Quick Recap

Cross Validation – Meaning and Need	• Process of evaluating the model on 'Out of Sample' data • Important because although a model is built on historical data, ultimately it is to be used on future data
Hold Out Validation	• Data is split into training and testing. Model developed on training and validated on testing