

Framingham Heart Study Data analysis

Main findings:

This data set was consisted of the following fields

```
'RANDID', 'SEX', 'TOTCHOL', 'AGE', 'SYSBP', 'DIABP',  
'CURSMOKE', 'CIGPDAY', 'BMI', 'DIAB', 'PREVCHD',  
'PREVAP', 'PREVMI', 'PREVSTRK', 'PREVHYP', 'TIME',  
'PERIOD', 'HDL', 'LDL', 'DEATH', 'ANGINA', 'HOSPMI',  
'MI_FCHD', 'ANYCHD', 'STROKE', 'CVD', 'HYPERTEN',  
'TIMEAP', 'TIMEMI', 'TIMEMIFC', 'TIMECHD', 'TIMESTRK',  
'TIMECVD', 'TIMEDTH', 'TIMEHYP'
```

Of which when these fields were taken in consideration for the project study I have come to this conclusion that:

LDL and HDL both are associated with CVD.

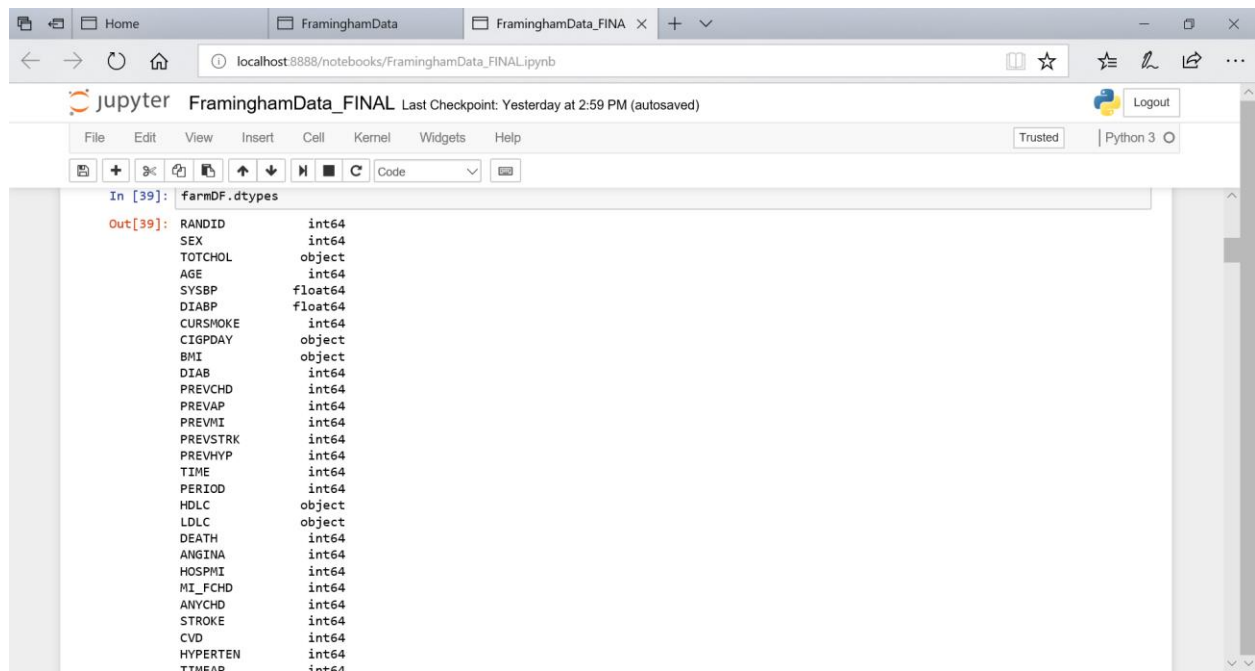
Other fields like DIAB, HYPERTEN, AGE, DIABP, SYSBP are also among some of the variables that have correlation/association with “CVD” apart from LDL and HDL.

And when applied logistic regression taking the above fields to predict the CVD the accuracy level was: 77 %

This could be enhanced by taking more fields into consideration.

Data exploration and Visualizations:

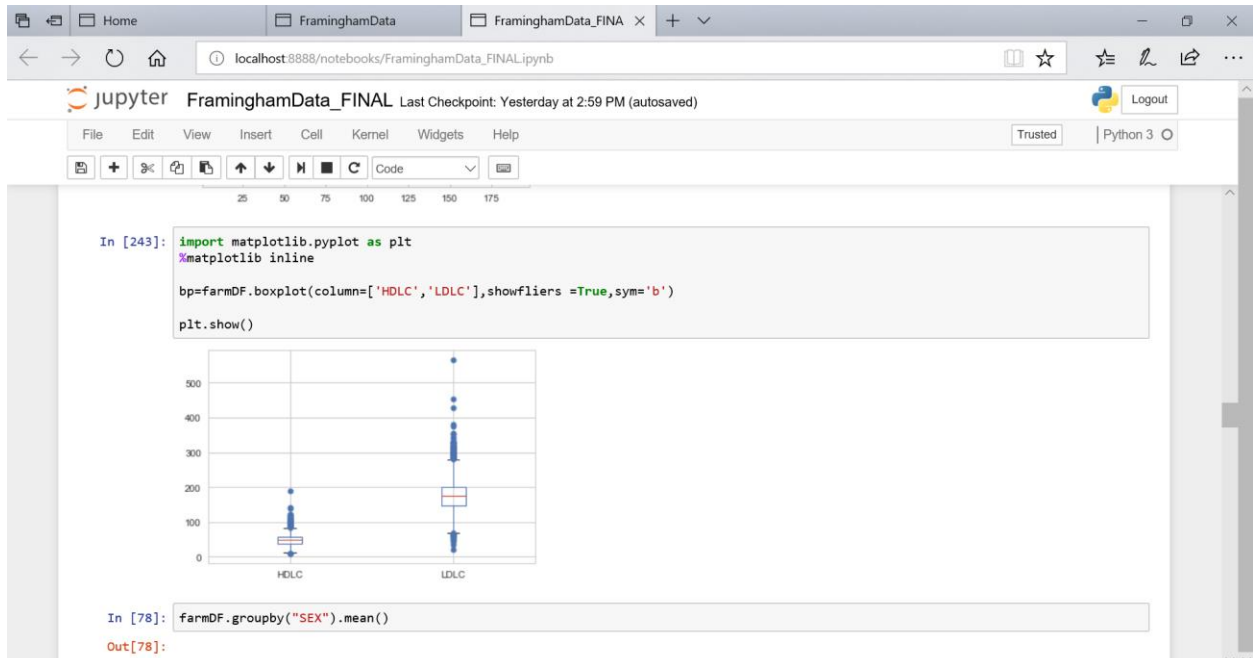
The datatype for the fields in the data set:



The screenshot shows a Jupyter Notebook interface with a browser window at the top displaying the URL `localhost:8888/notebooks/FraminghamData_FINAL.ipynb`. The notebook title is `FraminghamData_FINAL` and it shows the last checkpoint from yesterday at 2:59 PM. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, code execution, and output viewing. The current cell is a code cell with the input `farmDF.dtypes`. The output, labeled `Out[39]:`, displays the data types for each column in the `farmDF` DataFrame.

Field	Datatype
RANDID	int64
SEX	int64
TOTCHOL	object
AGE	int64
SYSBP	float64
DIABP	float64
CURSMOKE	int64
CIGPDAY	object
BMI	object
DIAB	int64
PREVCHD	int64
PREVAP	int64
PREVMI	int64
PREVSTRK	int64
PREVHYP	int64
TIME	int64
PERIOD	int64
HDL	object
LDL	object
DEATH	int64
ANGINA	int64
HOSPMI	int64
MI_FCHD	int64
ANYCHD	int64
STROKE	int64
CVD	int64
HYPERTEN	int64
TTMFAD	int64

Since there were missing values as well I replaced them (HDL and LDL) with their respective means.



Home FraminghamData FraminghamData_FINAL x + -

localhost:8888/notebooks/FraminghamData_FINAL.ipynb

jupyter FraminghamData_FINAL Last Checkpoint: Yesterday at 2:59 PM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [241]: #converting the datatype of null object to int and replacing the null values with mean
#set(farmDF['HDLc'].tolist())

farmDF['HDLc'].replace(0,nan, inplace=True)
farmDF["HDLc"]=farmDF["HDLc"].fillna(farmDF["HDLc"].mean())
farmDF1 = farmDF[['HDLc','LDLC']]

farmDF1.tail(10)
```

Out[241]:

	HDLc	LDLC
3224	49.398466	176.511007
3225	49.398466	176.511007
3226	49.398466	176.511007
3227	49.398466	176.511007
3228	49.398466	176.511007
3229	49.398466	176.511007
3230	49.398466	176.511007
3231	49.398466	176.511007
3232	49.398466	176.511007
3233	49.398466	176.511007

In [217]: #converting the datatype of LDLc (object to int) and replacing the null values with mean

The screenshot shows a Jupyter Notebook titled 'FraminghamData_FINAL'. The code in the cell processes the 'HDLC' variable by replacing null values with the mean and then selecting the first 10 rows of the resulting dataset. The output shows a table with two columns, HDLC and LDLC, and 10 rows of data.

```
In [241]: #converting the datatype of HDLC object to int and replacing the null values with mean
#set(farmDF['HDLC'].tolist())

farmDF['HDLC'].replace(0, nan, inplace=True)
farmDF["HDLC"] = farmDF["HDLC"].fillna(farmDF["HDLC"].mean())
farmDF1 = farmDF[['HDLC', 'LDLC']]

farmDF1.tail(10)
```

Out[241]:

	HDLC	LDLC
3224	49.398466	176.511007
3225	49.398466	176.511007
3226	49.398466	176.511007
3227	49.398466	176.511007
3228	49.398466	176.511007
3229	49.398466	176.511007
3230	49.398466	176.511007
3231	49.398466	176.511007
3232	49.398466	176.511007
3233	49.398466	176.511007

```
In [217]: #converting the datatype of LDLC (object to int) and replacing the null values with mean
```

Summary:

To check the association between the CVD and other fields I used statsmodels.api and calculated the summary with the variables.

The screenshot shows a Jupyter Notebook titled 'FraminghamData_FINAL'. The code fits a logit model and prints its summary. The output includes model statistics and a detailed table of coefficients for various health variables.

```
result=log_mod.fit()
print(result.summary())
```

Optimization terminated successfully.
Current function value: 0.493248
Iterations 6

Logit Regression Results

```
=====
```

Dep. Variable:	CVD	No. Observations:	2263
Model:	Logit	Df Residuals:	2256
Method:	MLE	Df Model:	6
Date:	Mon, 11 Dec 2017	Pseudo R-squ.:	0.07659
Time:	14:18:57	Log-Likelihood:	-1116.2
converged:	True	LL-Null:	-1208.8
		LLR p-value:	2.714e-37

```
=====
```

	coef	std err	z	P> z	[0.025	0.975]
HDLC	-0.0368	0.004	-9.568	0.000	-0.044	-0.029
LDLC	-0.0033	0.001	-2.916	0.004	-0.006	-0.001
DIAB	0.7768	0.172	4.522	0.000	0.440	1.114
HYPERTEN	0.4920	0.152	3.241	0.001	0.194	0.790
AGE	0.0187	0.005	3.452	0.001	0.008	0.029
DIABP	-0.0313	0.005	-6.063	0.000	-0.041	-0.021
SYSBP	0.0146	0.003	4.248	0.000	0.008	0.021

```
=====
```

```
In [227]: cols=["HDLC", "LDLC", "HYPERTEN", "AGE", "DIABP", "SYSBP", "DIAB"]
#cols=["HDLC", "LDLC"]
```

The p-values for almost all the variables are smaller than 0.05

Since, the p value is less than the significance value of 0.05 we have enough evidence to conclude that the field:

LDLC and HDLC associated with CVD

Also,

DIAB, HYPERTEN, AGE, DIABP, SYSBP are among some fields which are also significant for the model to predict the value of CVD.

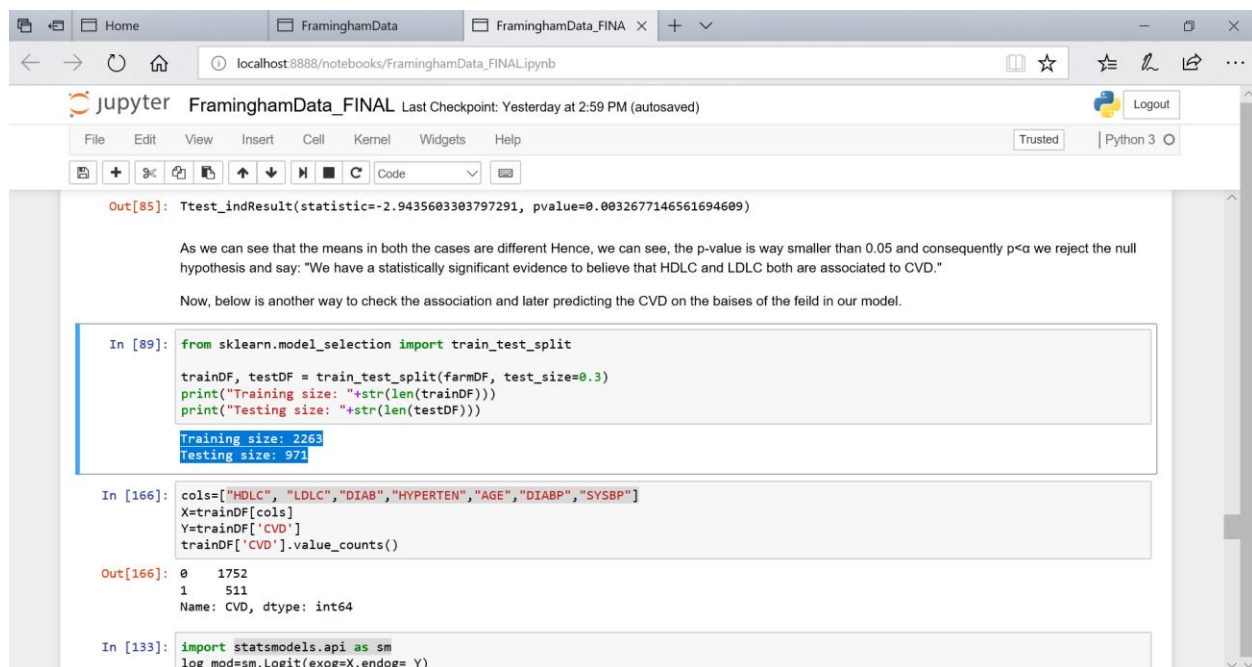
Testing and training dataset

Considering DIAB, HYPERTEN, AGE, DIABP, SYSBP, LDLC, HDLC all are significant and cannot be ignored for calculating/Predicting CVD.

For this purpose, I will split the dataset into two parts:

Test data: To test the accuracy of our model (30 percent of the data)

Training data: To train the model



The screenshot shows a Jupyter Notebook window titled "FraminghamData_FINAL". The browser address bar shows "localhost:8888/notebooks/FraminghamData_FINAL.ipynb". The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The code cell shows the following:

```
Out[85]: Ttest_indResult(statistic=-2.9435603303797291, pvalue=0.0032677146561694609)
```

As we can see that the means in both the cases are different Hence, we can see, the p-value is way smaller than 0.05 and consequently $p < \alpha$ we reject the null hypothesis and say: "We have a statistically significant evidence to believe that HDLC and LDLC both are associated to CVD."

Now, below is another way to check the association and later predicting the CVD on the baies of the feild in our model.

```
In [89]: from sklearn.model_selection import train_test_split

trainDF, testDF = train_test_split(farmDF, test_size=0.3)
print("Training size: "+str(len(trainDF)))
print("Testing size: "+str(len(testDF)))

Training size: 2263
Testing size: 971
```

```
In [166]: cols=["HDLC", "LDLC", "DIAB", "HYPERTEN", "AGE", "DIABP", "SYSBP"]
X=trainDF[cols]
Y=trainDF['CVD']
trainDF['CVD'].value_counts()
```

```
Out[166]: 0    1752
         1     511
         Name: CVD, dtype: int64
```

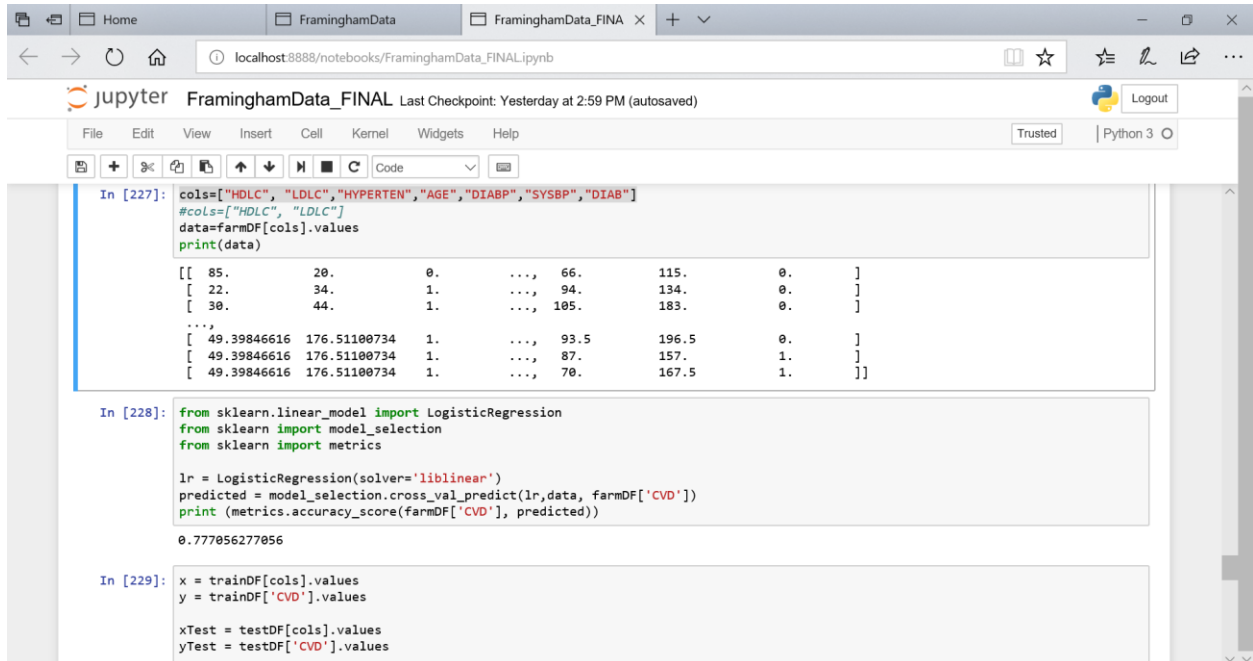
```
In [133]: import statsmodels.api as sm
log_mod=sm.Logit(exog=X, endog= Y)
```

Cross validation model selection

To check the fields give how much accuracy to the model we can also use cross validation method and then later using these variables in our model

I checked for all of these separately

```
cols=["HDL", "LDL", "HYPER", "AGE", "DIAB", "SYSBP", "DIAB"]
```



The screenshot shows a Jupyter Notebook interface with the following code cells:

```
In [227]: cols=["HDL", "LDL", "HYPER", "AGE", "DIAB", "SYSBP", "DIAB"]
#cols=["HDL", "LDL"]
data=farmDF[cols].values
print(data)
```

```
[[ 85.    20.     0.     ...  66.    115.     0.    ]
 [ 22.    34.     1.     ...  94.    134.     0.    ]
 [ 30.    44.     1.     ... 105.    183.     0.    ]
 ...,
 [ 49.39846616 176.51100734 1.     ...  93.5    196.5     0.    ]
 [ 49.39846616 176.51100734 1.     ...  87.     157.     1.    ]
 [ 49.39846616 176.51100734 1.     ...  70.    167.5     1.    ]]
```

```
In [228]: from sklearn.linear_model import LogisticRegression
from sklearn import model_selection
from sklearn import metrics

lr = LogisticRegression(solver='liblinear')
predicted = model_selection.cross_val_predict(lr,data, farmDF['CVD'])
print (metrics.accuracy_score(farmDF['CVD'], predicted))

0.777056277056
```

```
In [229]: x = trainDF[cols].values
y = trainDF['CVD'].values

xTest = testDF[cols].values
yTest = testDF['CVD'].values
```

and the accuracy was not more than 78 %.

Logistic Regression Model Fitting

After splitting the data set we can use our training data to train and further predict the values for the test data

The screenshot shows a Jupyter Notebook titled 'FraminghamData_FINAL' with a last checkpoint from yesterday at 2:59 PM. The notebook is running on a local host at 8888. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and output viewing. The code in the notebook is as follows:

```
print(metrics.accuracy_score(farmDF['CVD'], predicted))
0.777056277056

In [229]: x = trainDF[cols].values
          y = trainDF['CVD'].values

          xTest = testDF[cols].values
          yTest = testDF['CVD'].values

          lr.fit(x,y)
          print ("Coefficient: "+str(lr.coef_))
          print ("Intercept: "+str(lr.intercept_))
          print ("R-Squared: "+str(lr.score(xTest,yTest)))

          predicted = lr.predict(xTest)
          print (metrics.accuracy_score(yTest, predicted))

Coefficient: [[-0.02932182 -0.00084245  0.31816736  0.04775001 -0.0091127  0.01271408
  0.7808557 ]]
Intercept: [-4.01598851]
R-Squared: 0.775489186406
0.775489186406

In [237]: resultDF = testDF.assign(prediction=predicted)
          cols1=["CVD","prediction"]
          result=resultDF[cols1]
          result.head(30)
```

Accuracy score: 77 % (Which is not bad but can be made better by adding more variables in our model.)

Report by:

Jyotsana Yadav

Email: [jy82398n@pace.edu](mailto: jy82398n@pace.edu)

Course: Data Warehousing, Mining & Vis FALL 2017 IS-665

