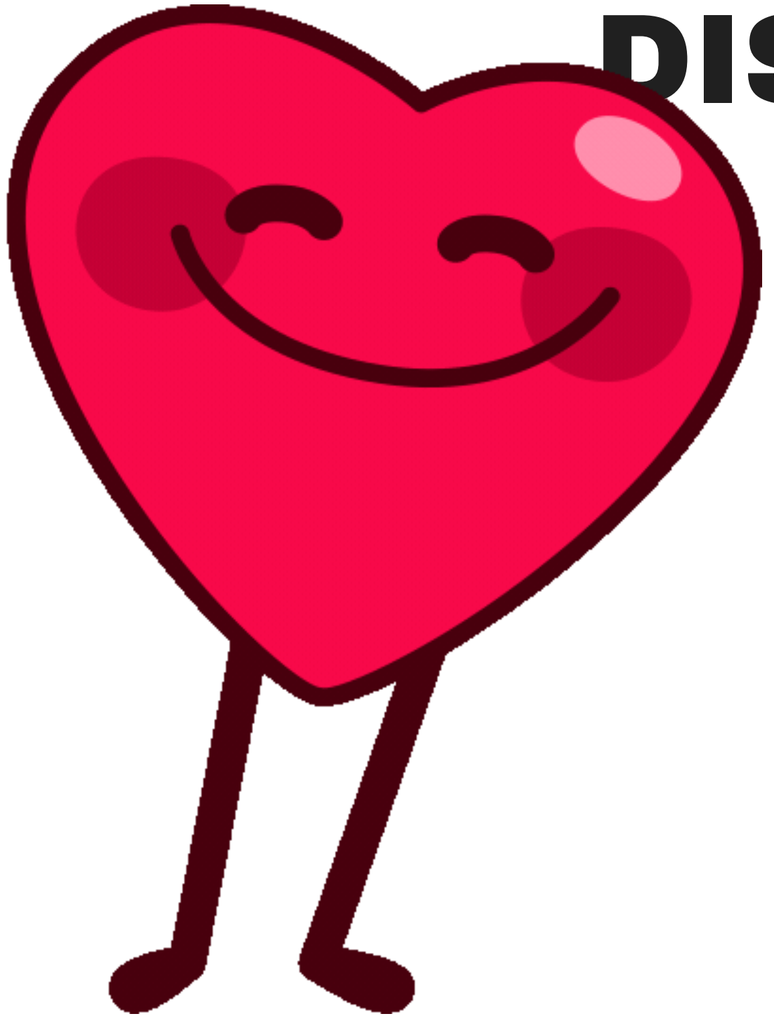
GRAPHIC ERA (DEEMED TO BE UNIVERSITY)

MACHINE LEARNING IN HEALTHCARE

CARDIOVASCULAR DISEASE PREDICTION



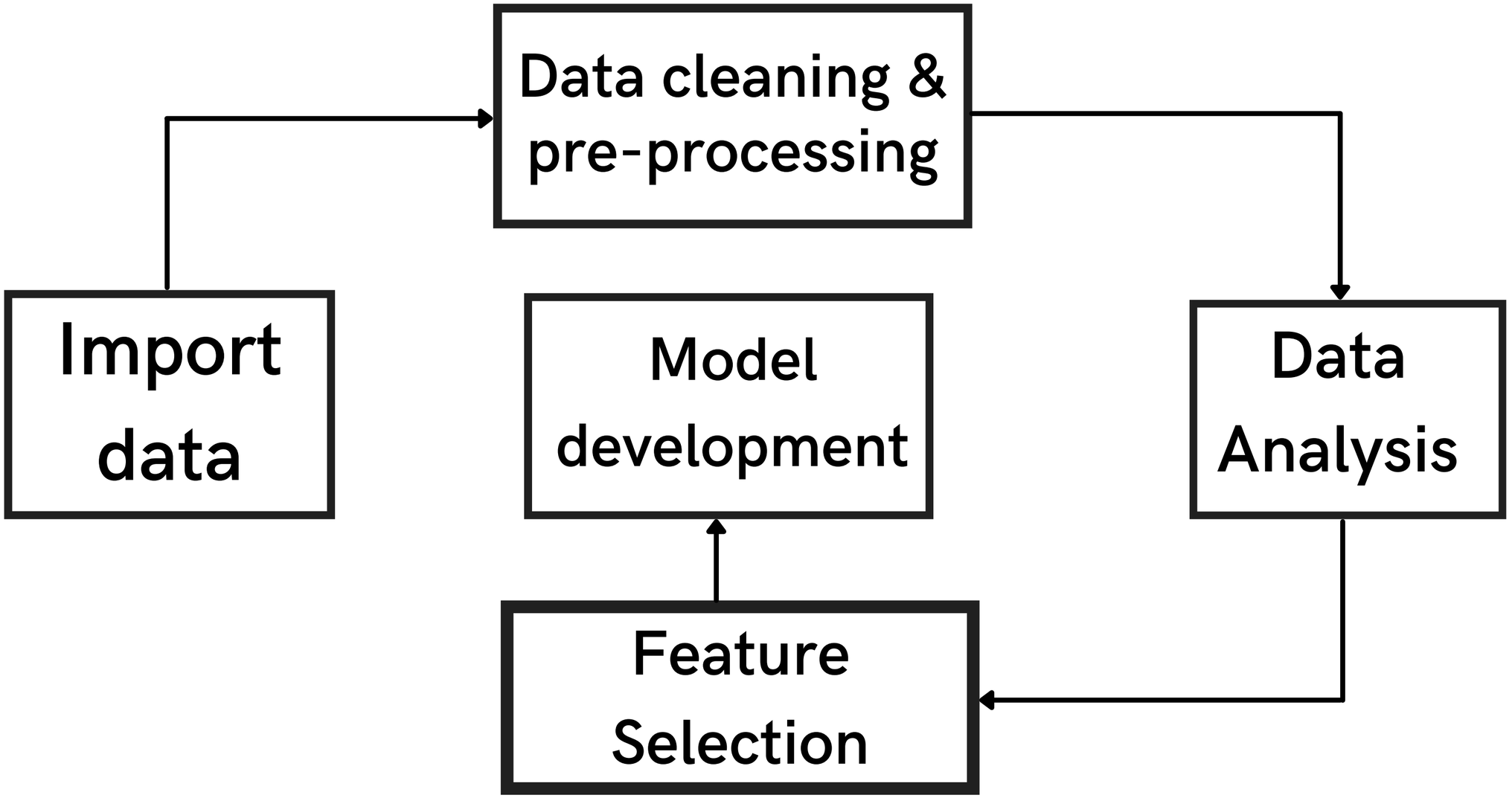
Prakhar Bhandari
Sec: ML
Sem. 4
University Roll No. - 2015258
Contact - +918126368858
Class Roll No. - 52
email - prakhar.luke@gmail.com

→

Cardiovascular disease is the leading cause of death worldwide accounting for one-third of deaths in 2019

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year. CVDs are a group of disorders of the heart and blood vessels and include coronary heart disease, cerebrovascular disease, rheumatic heart disease and other conditions. Four out of 5CVD deaths are due to heart attacks and strokes, and one third of these deaths occur prematurely in people under 70 years of age.

BLOCK DIAGRAM



LIBRARIES USED

numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices.

sklearn

Features various classification, regression and clustering algorithms including support vector machines

seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

for data manipulation & analysis
In particular, it offers data structures an&d operations for manipulating numerical tables & time series.

pandas

It provides a MATLAB-like way of plotting.

pyplot

Data cleaning and pre-processing:

Here I checked and dealt with missing and duplicate variables from the data set as these can grossly affect the performance of different machine learning algorithms (many algorithms do not tolerate missing data)

→

```
data = pd.read_csv('/kaggle/input/framingham-heart-study-dataset/framingham.csv')
data.drop(['education'],axis=1,inplace=True) #dropping the education column because it has no correlation with heart disease
data.head()
```

	male	age	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	TenYearCHD
0	1	39	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	0
1	0	46	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	0
2	1	48	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	0
3	0	61	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	1
4	0	46	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	0

Data Analysis:

Here I wanted to gain important statistical insights from the data and the things that I checked for were the distributions of the different attributes, correlations of the attributes with each other and the target variable and I calculated important odds and proportions for the categorical attributes.

```
#total percentage of missing data
missing_data = data.isnull().sum()
total_percentage = (missing_data.sum()/data.shape[0]) * 100
print(f'The total percentage of missing data is {round(total_percentage,2)}%')
```

```
The total percentage of missing data is 12.74%
```

Since the missing entries account for only 12% of the total data we can drop these entries without losing alot of data.

```
# drop missing entries
data.dropna(axis=0, inplace=True)
```

```
data.shape
```

```
(3751, 15)
```

Feature Selection:

Since having irrelevant features in a data set can decrease the accuracy of the models applied, I used the Boruta Feature Selection technique to select the most important features which were later used to build different models.

```
# show the most important features
most_important = data.columns[:-1][feat_selector.support_].tolist()
most_important
```

```
['age', 'sysBP']
```

We see that age and systolic blood pressures are selected as the most important features for most important features to build our models to get more features for more precise prediction.

```
# select the top 6 features
top_features = data.columns[:-1][feat_selector.ranking_ <=6].tolist()
top_features
```

```
['age', 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose']
```

The top features are:

1. Age
2. Total cholesterol
3. Systolic blood pressure
4. Diastolic blood pressure
5. BMI
6. Heart rate
7. Blood glucose

Model development

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import recall_score,precision_score,cla
```

```
# search for optimun parameters using gridsearch
params = {'penalty':['l1','l2'],
          'C':[0.01,0.1,1,10,100],
          'class_weight':['balanced',None]}
logistic_clf = GridSearchCV(LogisticRegression(),param_grid=
```

```
#train the classifier
logistic_clf.fit(X_train,y_train)

logistic_clf.best_params_
```

```
{'C': 10, 'class_weight': None, 'penalty': 'l2'}
```

Logistic Regression

Which models the probability of a data point belonging to a particular class and assigns this point the appropriate label based on a chosen threshold.

→

Accuracy:

```
#make predictions
logistic_predict = logistic_clf.predict(X_test)
```

```
log_accuracy = accuracy_score(y_test,logistic_predict)
print(f"Using logistic regression we get an accuracy of {
```

```
Using logistic regression we get an accuracy of 67.51%
```

→

Score / precision:

```
print(classification_report(y_test,logistic_predict))
```

```
              precision    recall  f1-score   support

           0       0.69      0.76      0.73       647
           1       0.65      0.56      0.60       498

    accuracy                           0.68      1145
   macro avg       0.67      0.66      0.66      1145
weighted avg       0.67      0.68      0.67      1145
```

```
logistic_f1 = f1_score(y_test, logistic_predict)
print(f'The f1 score for logistic regression is {round(logistic_f1*100,2)}%')
```

```
The f1 score for logistic regression is 60.09%
```

Conclusion

This model can then be used as a simple screening tool.

However, as a sanity check, most of the data on the positive cases were artificially synthesized using SMOTE and as such they may not be a true representation of the actual population data thus more data, especially on the positive cases, is needed to build better models and much more potent screening tools.

References:

KAGGLE

For Framingham dataset

WHO

For the essential information related to the topic

GITHUB

Different models and approach to solver some issues.

WIKIPEDIA

For all the info regarding cardiac disease and the factors resulting in heart attack.

PROF. ASHISH KANDERI (IIT-K)

For his guidance through this project and providing me with all the resources.

MR. UTKARSH KUSHWAHA

As one of the collaborator of this project he helped me a lot especially with the presentation.

Thank you!