# Predicting Meningitis Disease Outbreak in Nigeria

**Jeeva Mary Loui**

**Metro College Of Technology**

Predicting Meningits Disease Outbreak In Nigeria

Using Supervised Learning Classification Methods

Abstract

To find the best model to classify the outbreak of meningitis disease outbreak in Nigeria different classification methods are run, which are Logistic Regression , K-Nearest neeighbours , Random Forest , Adaboost and Support Vector Machine. Their performance are evaluated and the best model suited is found.

# Predicting Meningitis Disease Outbreak in Nigeria

## Dataset

Source: Kaggle.

Source url : https://www.kaggle.com/eiodelami/disease-outbreaks-in-nigeria-datasets

Meningitis is an inflammation of the membranes (meninges) surrounding your brain and spinal cord.It has claimed many lives in people all around the earth, predicting possible outbreaks could help minimize the next possible outbreaks and to maximize vaccine administration.

The dataset has 284484 rows and 40 columns.

## Data Preprocessing[1]

The size of datasetis so huge that it might cause very slow running of the classifiction techniques, so the size reduced by taking a sample of three percent of the total rows (dataset = dataset.sample(frac=0.01) )which reduced the size of dataset to (2845, 40).

The dataset is so clean that there are no null values and all the categorical values are hot encoded within the dataset itself. There are many columns which can be droped to get more clean data.

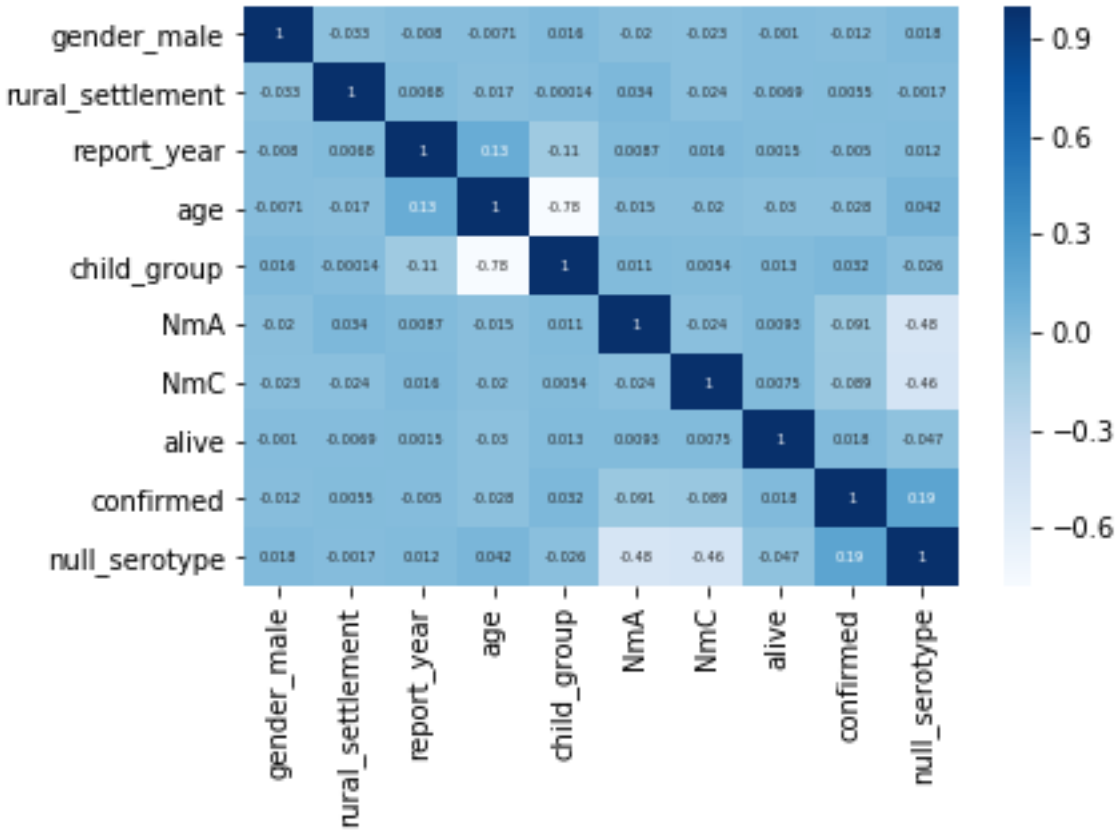
```
dat=dataset.drop(['id', 'surname', 'firstname', 'middlename', 'gender', 'gender_female', 'state', 'settlement', 'urban_settlement', 'report_date', 'age_str', 'date_of_birth','adult_group', 'cholera', 'diarrhoea', 'measles', 'viral_haemmorrhaphic_fever', 'meningitis', 'ebola', 'marburg_virus', 'yellow_fever', 'rubella_mars', 'malaria', 'serotype','NmW', 'health_status', 'dead', 'report_outcome', 'unconfirmed'],axis=1)
```

Our new dataset is saved to ‘dat’.The target variable is disease which is categorical and is named to numerical classes using cat.codes.

```
dat['disease'] = dat['disease'].astype('category').cat.codes
```

**Correlation Analysis.**

For succesful classification the features with strong correlation with target varible can be selcted using correlation analysis.Seaborn heatmap is an efficient method to study the correlation between variables.




The features having high correlations are represented with dark blue color as the color scale depicted in the heatmap.

**Recursive Feature Elimination.**

Many Features can be elimated by using RFE which is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached.

Fisrst the dataset is split to feature set and target dataframe.

```
target=dat['disease']
X=dat.drop(['disease'], axis=1)
y=target
```

The optimal number of features is found to be two and they are 'confirmed' and 'null_serotype'. Null sero type is null for unknown meningitis serogroup or for diseases other than meningitis and Confirmed is report outcome of confirmed disease.

```
X_sub = X.loc[:,selected_features]
X_train, X_test, y_train, y_test = train_test_split(X_sub,y,random_state=0)
```

The features and target are split to test and train sets to run the model and evaluate it.

**Logistic Regression for Classification.**

From scikit learn library LogisticRegression classifier is imported and the model is fit with the training set.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)  (Last Name, Year)
```

The model is evaluated to get an accuracy score of 0.24578651685393257

And the classification report is:

recall(0.136660522063676 3, 0.2488372093023255 8, 0.1561036468330134 2, None)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Cholera | 0.00 | 0.00 | 0.00 | 68 |
| Diarrhoea | 0.22 | 0.49 | 0.30 | 86 |
| Ebola | 0.15 | 1.00 | 0.26 | 68 |
| Malaria | 0.00 | 0.00 | 0.00 | 69 |
| Marburg Virus | 0.00 | 0.00 | 0.00 | 67 |
| Measles | 0.00 | 0.00 | 0.00 | 81 |
| Meningitis | 1.00 | 1.00 | 1.00 | 65 |
| Rubella Mars | 0.00 | 0.00 | 0.00 | 64 |
| Viral Haemmorrhaphic Fever | 0.00 | 0.00 | 0.00 | 65 |
| Yellow Fever | 0.00 | 0.00 | 0.00 | 79 |
| micro avg | 0.25 | 0.25 | 0.25 | 712 |
| macro avg | 0.14 | 0.25 | 0.16 | 712 |
| weighted avg | 0.13 | 0.25 | 0.15 | 712 |

*Kfold Cross Validation*

In K Fold cross validation, the data is divided into k subsets and train our model on k-1 subsets and hold the last one for test. This process is repeated k times, such that each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set. We then average the model against each of the folds and then finalize our model.

Doing K fold cross validation with logistic regression accuracy of : .2428

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kf = KFold(n_splits=10, shuffle=True, random_state=1)
result=cross_val_score(model,X,y,cv=kf)
```

Thus the accuracy is same with and without K-Fold Cross Validation.

**K-Nearest NeighborsClassifier**

Model is trained using KneighborsClassifier. And the accuracy score is .2457 and confusion matrix is same for Logistic Regression as well.

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
model.fit(X_train,y_train)
acc_score.update(KNeighborsClassifier = perf(y_test,y_pred))
```

*Grid Search Cross Validation:*

To check the improvement of accuracy with cross validation Grid Search Cross Validation is used.

```
from sklearn.model_selection import GridSearchCV
```

For K-nearest neighbours the best parameters are found to be,

```
model.best_params_
{'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'distance'}
```

From parameters:

```
param_dict = {
        'n_neighbors':[3,5,7,4,11],
        'weights':['uniform','distance'] ,
```

```
                'metric':['euclidean','manhattan']
            }
```

The model's best score is **:** 0.1796133567662566 ,Which implies grid search donot improve the result for this model. Also it is found that

```
train_scores.mean()
0.6716315572254813
test_scores.mean()
0.15694200351493848
```

The train score is very large that test score which implies strong overfitting.

**Adaboost For Classification**

Adaptive boosting is an ensemble method used for classification.

From sklearn.ensemble library Adaboost Forest Classifier is imported and the model is run.

```
abc = AdaBoostClassifier(n_estimators=50,
                         learning_rate=1)
# Train Adaboost Classifer
model = abc.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = model.predict(X_test)
acc_score.update( AdaBoostClassifier= perf(y_test,y_pred)
```

The accuracy score is .2289 and the macro precion recall support values are :

```
recall(0.130474955053367, 0.24411764705882355, 0.14900441019179206, None)
                           precision    recall  f1-score   support

                  Cholera       0.15      0.44      0.23        68
                Diarrhoea       0.00      0.00      0.00        86
                    Ebola       0.15      1.00      0.26        68
                  Malaria       0.00      0.00      0.00        69
             Marburg Virus       0.00      0.00      0.00        67
                  Measles       0.00      0.00      0.00        81
                Meningitis       1.00      1.00      1.00        65
              Rubella Mars       0.00      0.00      0.00        64
Viral Haemmorrhaphic Fever       0.00      0.00      0.00        65
              Yellow Fever       0.00      0.00      0.00        79

                 micro avg       0.23      0.23      0.23       712
                 macro avg       0.13      0.24      0.15       712
              weighted avg       0.12      0.23      0.14       712
```

*Grid Search Cross Validation:*

The adaboost classifier is run with gridserach cross validation with cross validation equal to 4.

```
model = GridSearchCV(AdaBoostClassifier(), param_grid=param_dict,cv=4)
model.fit(X,y)
```

model.best_params_
{'learning_rate': 1, 'n_estimators': 20}
model.best_score_
0.2601054481546573
train_scores.mean()
0.23233090300310064
test_scores.mean()
0.22474516695957822

The model best score is .26 which is fine and also the train score is only a little greater than test score score so there isno much problem of overfitting

**Random Forest Classification**

Random forest is an ensemble method used for classification and regression .

From sklearn.ensemble library Random Forest Classifier is imported and the model is run.

```
rf = RandomForestClassifier(n_estimators=15, max_depth=3)
rf.fit(X_train,y_train)
rf.score(X_test,y_test)
y_pred = model.predict(X_test)
acc_score.update( RandomForestClassifier= perf(y_test,y_pred))
```

The accuracy score is . 0.2457 and the macro precion recall support values are :

```
recall(0.1366605220636763, 0.24883720930232558, 0.15610364683301342, None)
                           precision    recall  f1-score   support

                  Cholera       0.00      0.00      0.00        68
                Diarrhoea       0.22      0.49      0.30        86
                    Ebola       0.15      1.00      0.26        68
                  Malaria       0.00      0.00      0.00        69
             Marburg Virus       0.00      0.00      0.00        67
                  Measles       0.00      0.00      0.00        81
                Meningitis       1.00      1.00      1.00        65
              Rubella Mars       0.00      0.00      0.00        64
Viral Haemmorrhaphic Fever       0.00      0.00      0.00        65
              Yellow Fever       0.00      0.00      0.00        79

                 micro avg       0.25      0.25      0.25       712
                 macro avg       0.14      0.25      0.16       712
              weighted avg       0.13      0.25      0.15       712
```

*Grid Search CV*

Running Grid Search CV for ranndom forest

model.best_score_

0.2616033755274262

train_scores.mean()

0.2681050499874033

```
test_scores = model.cv_results_['mean_test_score']
test_scores.mean()
0.26085325832161277
```

The model best score is .26160 and also the train and test scores are equal . Hence Random forest could possibly be a good model.

**Support Vector Machine for Classification**

**SVM** is another method for classification. From Scikit learn library SVM is imported and Support Vector Classifier is used to train a classification model.

```
from sklearn.svm import SVC
clf = SVC()
clf.fit(X_train,y_train)
```

The accuracy score is 0.24578651685393257 and the macro precion recall support values are:

```
recall(0.1366605220636763, 0.24883720930232558, 0.15610364683301342, None)
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Cholera | 0.00 | 0.00 | 0.00 | 68 |
| Diarrhoea | 0.22 | 0.49 | 0.30 | 86 |
| Ebola | 0.15 | 1.00 | 0.26 | 68 |
| Malaria | 0.00 | 0.00 | 0.00 | 69 |
| Marburg Virus | 0.00 | 0.00 | 0.00 | 67 |
| Measles | 0.00 | 0.00 | 0.00 | 81 |
| Meningitis | 1.00 | 1.00 | 1.00 | 65 |
| Rubella Mars | 0.00 | 0.00 | 0.00 | 64 |
| Viral Haemmorrhaphic Fever | 0.00 | 0.00 | 0.00 | 65 |
| Yellow Fever | 0.00 | 0.00 | 0.00 | 79 |
| micro avg | 0.25 | 0.25 | 0.25 | 712 |
| macro avg | 0.14 | 0.25 | 0.16 | 712 |
| weighted avg | 0.13 | 0.25 | 0.15 | 712 |

*GridSearch CV*

Doing Grid Search Cross validation for SVM ,

```
model.best_score_
0.26629160806375995
train_scores.mean()
.21578006513837195
test_scores.mean()
0.2121685680054175
model.best_params_
```

{'C': 0.1, 'degree': 2, 'kernel': 'rbf'}

The model best score is .266 and the train score and test score are almost equal thus there are no signs of overfiting.

## CONCLUSION

Comparing the accuracy score of all different model the best is RandomForest Classifier with GridSearch cross validation. It shows no sign of overfitting with best score of .261.