UE17CS303 - Machine Learning Project - Prediction of Low birth Weight Cases

Pradyumna Y M - PES1201700986
Anush V Kini - PES1201701646
Vikas Matam - PES1201701665

October 29, 2020

1 Problem Statement

Low Birth weight (LBW) acts as an indicator of sickness in newborn babies. LBW is closely associated with infant mortality as well as various health outcomes later in life. Various studies show strong correlation between maternal health during pregnancy and the child's birth weight. Exploit machine learning techniques to gain useful information from health indicators of pregnant women for early detection of potential LBW cases.

The forecasting problem should be reformulated as a classification problem between LBW and NOT-LBW classes.

2 Data Cleaning

Given data has 10 columns:

- **community**: This column corresponds to the community the mother belongs to. It is a categorical variable that takes values from 1 to 4.
- **age**: Age of the Mother
- **weight1**: Avg Weight of the mother in Kg
- **history**:A categorical numeric data which takes value between 1 and 6
- **HB**:This represents Hemoglobin concentration in g/l.
- **IFA**:Categorical variable whether the mother took folic acid or not.
- **BP1**:Average BP of the mother during first trimester
- **education**:Variable which represents the level education of mother
- **res**:Column represents if the mother is a resident of a village or not.
- **reslt**:This column represents is the born baby is under weight or not.

There were a number of missing values in the dataset. After examining the dataset, we decided to impute the missing values based on the means after grouping the data points based on the target variable 'reslt'.

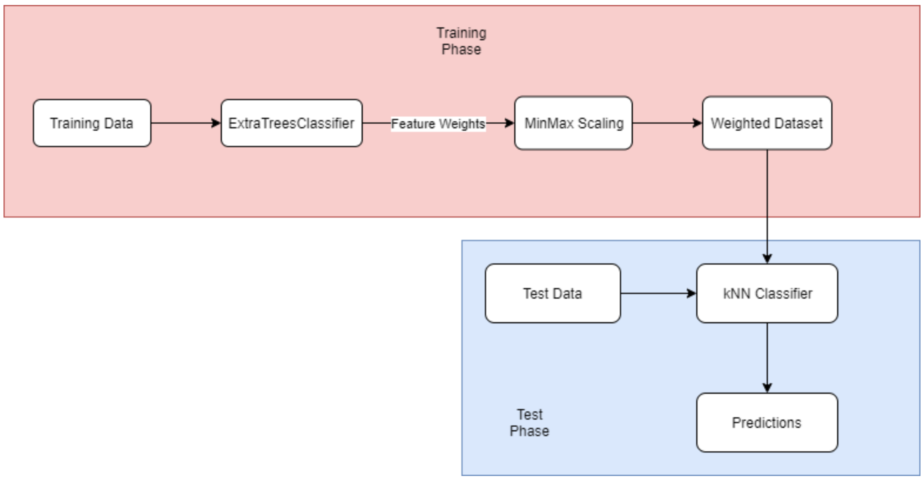

Figure 1: The proposed Pipeline

3 Methodology

After cleaning the dataset, we tried out a wide range of models to serve as baseline accuracies to beat. These baseline accuracies have been tabulated in the results section.

We noticed that most of the models were performing poorly due to the curse of dimensionality. So, we came up with two methods of assigning a weight to each column in the dataset.

3.1 PCA

To overcome the curse of dimensionality, we propose a method of applying PCA on the dataset as a feature selector. PCA is applied on the dataset and the importance of different components is analysed. We select components such that the components explain about 95% of the variance in the dataset. The importance of each component has been visualised in Fig 2.

We applied PCA on the dataset, and noticed that only 3 of the principle components explained significant amount of the variance of the dataset. So, we decided to drop the remaining principal components. We then trained a kNN classifer and obtained a 5-fold cross validation accuracy of 90% and an test train split accuracy of 92%. A major disadvantage of this method is that we cannot explain what each PCA component is measuring.

3.2 Feature Weighting method

We noticed that some of the features, such as education and history did not contribute to the variance of the model. So, we decided to weigh the features based on their importances. For this, we used the ExtraTreesClassifier from sklearn . Once we got the weights, we normalized the dataset using a MinMax scaler and then multiplied the columns with their respective weights.
This method ensured that the columns which contribute more to the dataset are weighted higher. For example, according to the feature importances visualised

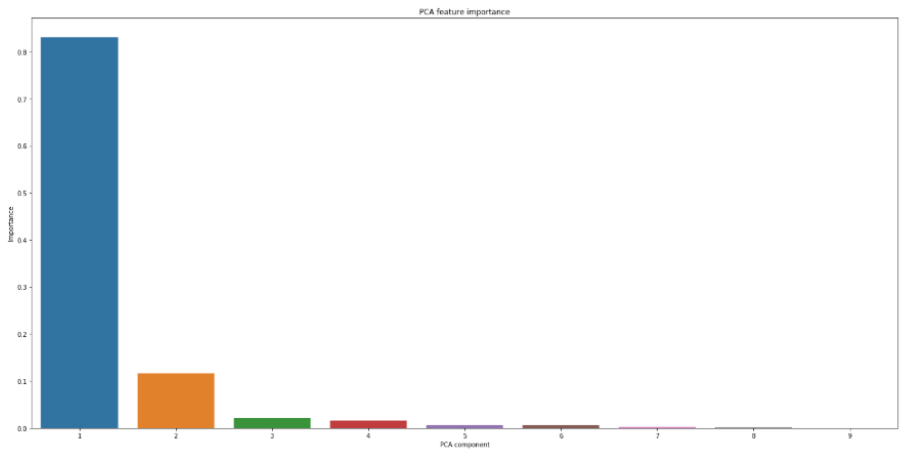

Figure 2: Importances of components from PCA

in Fig 4, the 'weight1' column has been assigned a feature importance of about 46%. This high importance can be attributed to the fact that the 'weight1' column determines the result in most of the cases. This can be seen in the scatter plot in Fig 3 After comparing the models based on their 5 fold cross validation accuracies, we decided to implement the kNN model, as it seemed to perform the best. Euclidean distance was used for the kNN model, and after examining the k vs Accuracy plot, we used k=7. kNN performed better than the other models because it was the only instance based learning algorithm. Due to the small size of the dataset, other models were not able to generalize well.

3.2.1 Extremely Randomized Trees Classifier

Extra Trees Classifier is an ensemble learning model is an ensemble learning meta algorithm similar to random forests. The only difference from a random forest classifier is that an extra tree classifier chooses splits randomly instead of choosing the split which minimises an impurity measure such as gini index. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.
Extra trees usually perform better than random forest when there is noise in the features. Hence, extra trees was used instead of random forest classifier.

Algorithm 1: The KNN algorithm

```
Result: Predictions of the target variable
predictions = list();
for ( each row in the test set ) {
    Sort all training instances based on distance from row
    l = target variable of the first k elements
    pred = mode(l)
    predictions.append(pred)
}
```

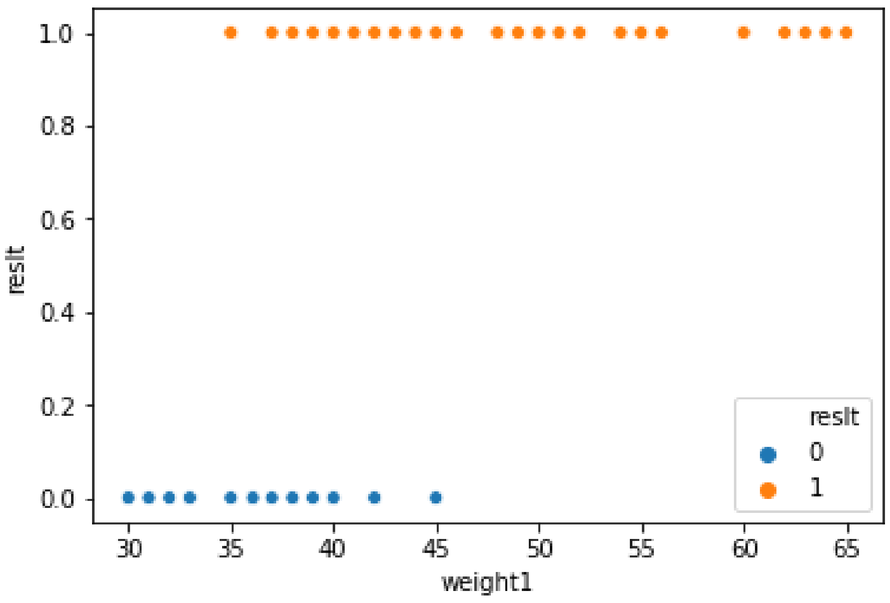

Figure 3: Scatterplot of weight1 vs reslt

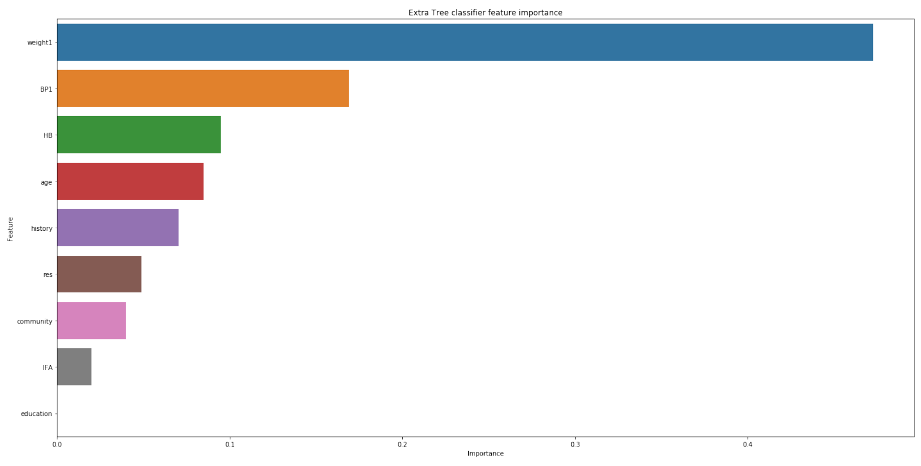

Figure 4: Importances of weights from ExtraTreesClassifier

4 Results and Conclusion

We obtained an accuracy of 96% and a 5 fold cross validation accuracy of 92.1% using our kNN model and the feature engineering described above.The final pipeline visualization can be seein in Fig 1

Table 1: A comparison of all approaches and their 5-Fold accuracies

Model	**Baseline**	**PCA**	**Feature Weighting**
GBDT	87.095	85.04	87.14
Random Forest	88.04	84.04	84.09
Adaboost	88.04	84.04	84.09
DNN	77.14	83.19	84.28
kNN	76.14	90.04	92.1
Radial SVM	81.04	93.99	78.23