LOVELY PROFESSIONAL UNIVERSITY

MATERNAL HEALTH RISK PREDICTION

Name - Mohit Kumar Mahato

Reg no - 11913514

Section - KM011

Roll no - RKM011B63

Course code – INT 417

Course name – Machine Learning Algorithm

MATERNAL HEALTH RISK PREDICTION

Mohit Kumar Mahato

Lovely Professional University, India

Abstract

This paper studies multiple machine learning models to predict the maternal health risk level of women. Maternal health means the health of women during pregnancy, childbirth and after childbirth. Each stage should be a positive experience, ensuring that women and their children reach their full potential in health and wellness. Although significant progress has been made in the past two decades, an estimated 295,000 women died during pregnancy and after childbirth in 2017. This figure is unacceptably high. Most maternal deaths are avoided by early treatment by a competent health professional working in a supportive environment.

The dataset used for the prediction is from the website archive.ics.uci.edu which is a basically a repository of machine learning datasets. According to a problem-solving approach, I have split it to 5 parts (Data understanding and exploration, Data preparation: Feature Engineering and Scaling and Model Building). I have used multiple models (Logistic Regression, Decision Tree Classifier, Random Forest Classifier, GaussianNB, Support Vector Classifier, K Nearest Neighbours) to predict the target variable to find out the model which is having the highest accuracy among all the models for this dataset.

Keywords: *Maternal health risk, childbirth, data understanding, Liner Regression, Decision Tree Classifier, Random Forest Classifier, GaussianNB, Support Vector Classifier, K Nearest Neighbours.*

1. Introduction

In this paper we will predict the maternal health risk of women using different machine learning models and then find out which model is giving having the highest accuracy and giving the best results. In addition to have an accurate estimation of the risk level we need to consider many factors, so the models depend on several different features and variables. Features used in for training are :-

- *Age*
- *Systolic Blood Pressure*
- *Diastolic Blood Pressure*
- *Blood Sugar*
- *Body Temperature*
- *Heart rate*

1.1 Need and application of this project

This project is important for decreasing the annual number of deaths during pregnancy because if the doctors know the risk level of the patient beforehand by using the model, then they can take the necessary actions to prevent any complications during pregnancy because as we all know that sooner the problem is known, more will be the chances to prevent it.

The doctors can collect all the required data needed by the model by performing required tests on the patients and then use all those data to predict the risk level from the machine learning model.

2. Literature Review

- *Emily F. Hamilton, Alina Dyachenko, et al. 2018 [1] determined whether machine learning strategies can detect specific clusters of conditions with varying probability estimates for severe neonatal morbidity and compare these findings to those based on original multivariable analysis. They concluded that by using a relatively small dataset and only few important factors that can be found out before birth it is possible to get more accurate estimate of the risk for severe neonatal morbidity on which clinicians can superimpose their medical judgment, experience, and intuition.*

- *Zeeshan Ahmed, Khalid Mohamed, et al. 2020[2] used Artificial intelligence with multi-functional machine learning platform development for better healthcare and precision medicine. They believe that the cutting-edge, new AI and ML-based big data platform development has the potential to revolutionize the field of medicine and improve the quality and transition of healthcare by intelligently analysing structured clinical data available in great count and volume, posing unprecedented challenges in data storage, processing, exchange, and curation, and developing a better understanding of biology.*

- *Lena Davidson & Mary Regina Boland 2020[3] demonstrated how AI has been applied to address pharmacological exposures during pregnancy and this includes the entire pregnancy process: preconception, prenatal, perinatal, and postnatal health concerns. They identified three areas where AI methods could be used to improve our understanding of pharmacological effects of pregnancy, including: a.) obtaining sound and reliable data from clinical records, b.) designing optimized animal experiments to validate specific hypotheses, c.) implementing decision support systems that inform decision-making.*

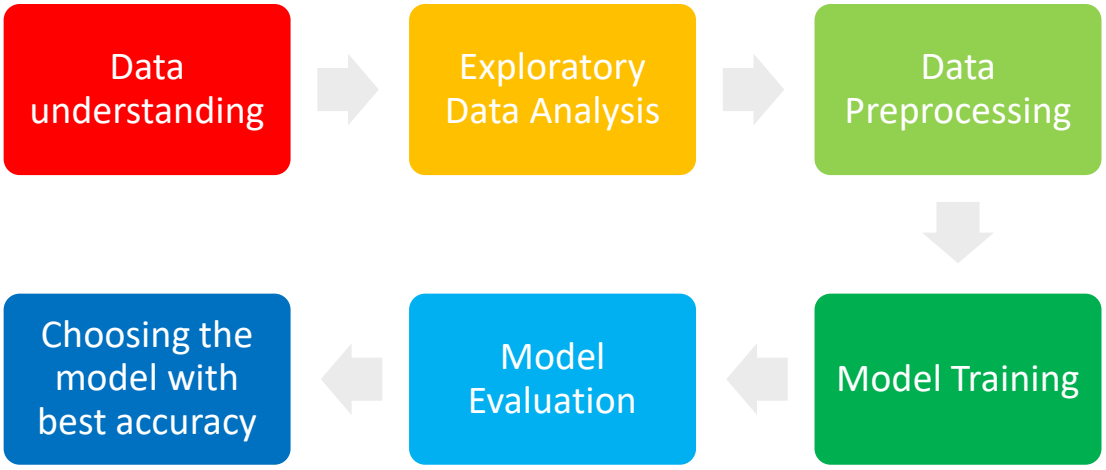
- *Ayleen Bertini, Rodrigo Salas, et al, 2022[4] identified the applicability and performance of machine learning methods used to identify pregnancy complications. They used multiple machine learning methods and got the best results of prediction of prematurity from medical images using the support vector machine technique, with an accuracy of 95.7%, and the prediction of neonatal mortality with the XGBoost technique, with 99.7% accuracy.*

- *Mary Regina Boland, Fernanda Polubriaginof, et al, 2017[5] developed a machine learning algorithm to classify drugs of unknown fetal effect. This will provide both pharmacologists and physicians with a much-needed initial classification of these 'unknown fetal effect' drugs.*

They used Logistic regression and random forest classifier to predict the classes of drugs of unknow fetal effects

3. Proposed Methodology

Figure 1- Steps followed in this project



3.1 Data Understanding

First, we will understand the attributes of the data, see which attributes are continuous and which are categorical. Then understand the problems with the data, such as missing values, inaccuracies, and outliers.

Figure 2- First five rows of the dataset

```
In [106]: data.head()
```

Out[106]:

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
0	25	130	80	15.0	98.0	86	high risk
1	35	140	90	13.0	98.0	70	high risk
2	29	90	70	8.0	100.0	80	high risk
3	30	140	85	7.0	98.0	70	high risk
4	35	120	60	6.1	98.0	76	low risk

In this dataset the attribute 'RiskLevel' is the target variable also known as the dependent variable and all the other features are the independent variables which will be used to train the model so that it can predict the RiskLevel on its own.

There are a total of 1014 rows in this dataset which means it has the data of 1014 women. The independent features of this dataset are: -

- *Age (in years)*
- *Systolic Blood Pressure*
- *Diastolic Blood Pressure*
- *Blood Sugar*
- *Body Temperature (in Fahrenheit)*
- *Heart rate (in BPM)*

Figure 3- Checking the data types of the attributes

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1014 entries, 0 to 1013
Data columns (total 7 columns):
Age            1014 non-null int64
SystolicBP     1014 non-null int64
DiastolicBP    1014 non-null int64
BS             1014 non-null float64
BodyTemp       1014 non-null float64
HeartRate      1014 non-null int64
RiskLevel      1014 non-null object
dtypes: float64(2), int64(4), object(1)
memory usage: 55.5+ KB
```

As we can see that the data types of all the independent variables are fine, and these variables can be directly used to train the model. The only change we will do to the dataset is that we will convert the datatype of the target variable from object to int64 by assigning the three categorical values an integer value (1: low, 2: medium, 3: high).

Figure 4- After changing the values of RiskLevel

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1014 entries, 0 to 1013
Data columns (total 7 columns):
Age            1014 non-null int64
SystolicBP     1014 non-null int64
DiastolicBP    1014 non-null int64
BS             1014 non-null float64
BodyTemp       1014 non-null float64
HeartRate      1014 non-null int64
RiskLevel      1014 non-null int64
dtypes: float64(2), int64(5)
memory usage: 55.5 KB
```

Now we will scan the data set for null values

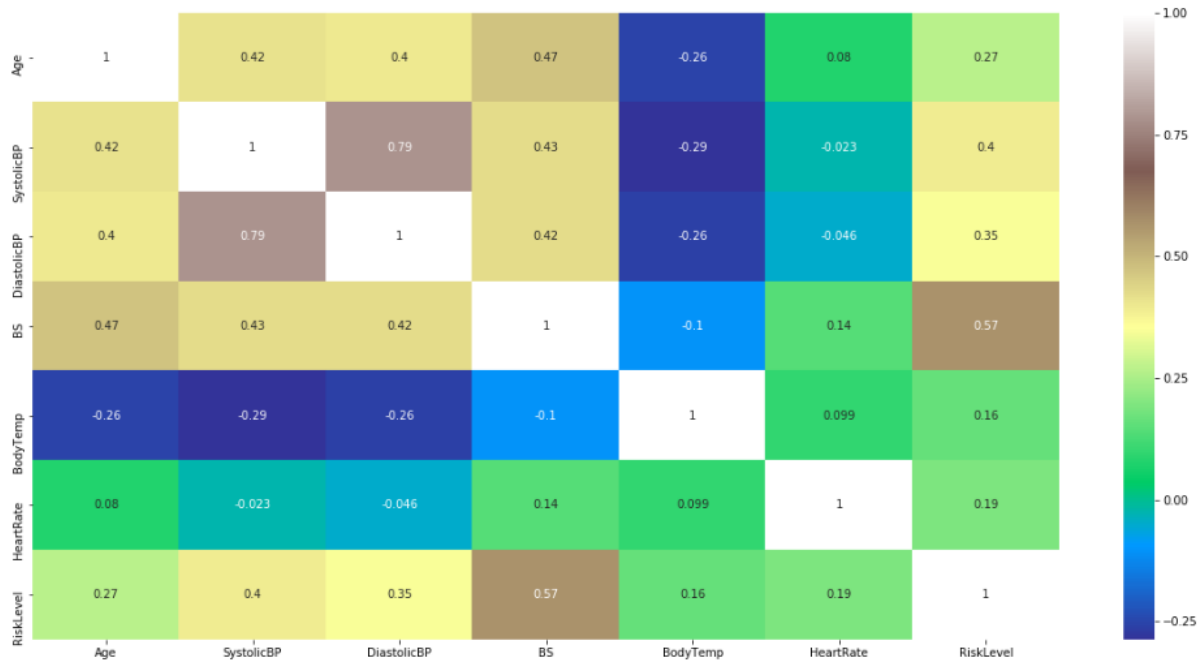
Figure 5- Checking NULL values in the dataset

```
In [113]: data.isnull().sum()

Out[113]: Age            0
          SystolicBP     0
          DiastolicBP    0
          BS             0
          BodyTemp       0
          HeartRate      0
          RiskLevel      0
          dtype: int64
```

There are no null values in the dataset so we can skip the data cleaning step and go to EDA.

Figure 6- Correlation heatmap



The correlation heatmap shows us that there is a positive correlation between the target variable and all the other independent variables.

3.2 Data Pre-processing

- ***Scaling the data***

```
In [117]: from sklearn.model_selection import train_test_split
          from sklearn.preprocessing import StandardScaler
          StandardScaler = StandardScaler()
          IC=['Age','SystolicBP','DiastolicBP','BS','BodyTemp','HeartRate']
          data[IC]= StandardScaler.fit_transform(data[IC])
```

```
In [118]: data.head()
```

Out[118]:

	Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel
0	-0.361738	0.913396	0.255023	1.905890	-0.485215	1.446956	3
1	0.380777	1.457027	0.975539	1.298340	-0.485215	-0.532088	3
2	-0.064732	-1.261127	-0.465493	-0.220537	0.973884	0.704815	3
3	0.009519	1.457027	0.615281	-0.524312	-0.485215	-0.532088	3
4	0.380777	0.369765	-1.186009	-0.797710	-0.485215	0.210054	1

Scaling means that we are transforming the data so that it fits within a specific scale, like 0-100 or 0-1. We want to scale the data because we are going to use methods based on measures of how far apart data points are, like SVM and KNN.

- ***Train – Test split***

```
X= data.drop(['RiskLevel'], axis=1)
Y= data['RiskLevel']
```

```
x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=0.30, random_state=50)
```

Here we are splitting the data into two parts – train and test, train part will be used for training the model and the test part will be used for testing the model and find how well the model is predicting the values.

```
print('X_train-', x_train.size)
print('X_test-',x_test.size)
print('y_train-', y_train.size)
print('y_test-', y_test.size)
```

```
X_train- 4254
X_test- 1830
y_train- 709
y_test- 305
```

We split the dataset into 70:30 ratio so 70% of the data will be used for training and 30% will be used for testing.

3.3 Model Training

3.3.1 Logistic regression

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Figure 6- Training logistic regression model

```
In [122]: from sklearn.linear_model import LogisticRegression
          lr=LogisticRegression()

          model1=lr.fit(x_train,y_train)
          prediction1=model1.predict(x_test)
```

Figure 7- Evaluation of logistic regression model

```
In [124]: sns.heatmap(cm, annot=True,cmap='BuPu')
Out[124]: <matplotlib.axes._subplots.AxesSubplot at 0x25bcd30ce48>
```


```
In [125]: accuracy_score(y_train, train_pred)
Out[125]: 0.5881523272214386

In [126]: print(classification_report(y_test, prediction1))
```

	precision	recall	f1-score	support
1	0.66	0.84	0.74	128
2	0.52	0.35	0.42	100
3	0.76	0.74	0.75	77
avg / total	0.64	0.65	0.64	305

From figure 7 we can see that we got an accuracy of 58% from the logistic regression model.

3.3.2 Decision Tree classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules, and each leaf node represents the outcome.

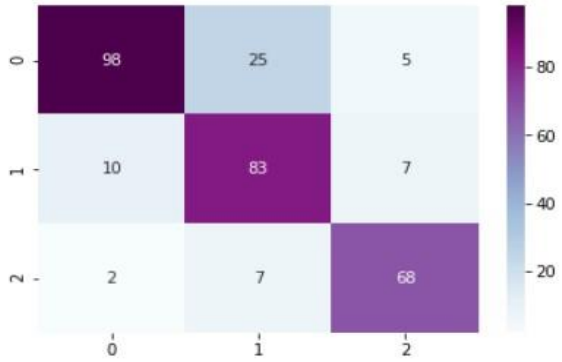
Figure 8- Training decision tree classifier model

```
In [127]: from sklearn.tree import DecisionTreeClassifier

          dtc=DecisionTreeClassifier()
          model2=dtc.fit(x_train,y_train)
          prediction2=model2.predict(x_test)
```

Figure 9- evaluating the model

```
In [129]: sns.heatmap(cm2, annot=True,cmap='BuPu')
Out[129]: <matplotlib.axes._subplots.AxesSubplot at 0x25bcd042588>
```




```
In [130]: accuracy_score(y_test,prediction2)
Out[130]: 0.8163934426229508

In [131]: print(classification_report(y_test, prediction2))
              precision    recall  f1-score   support

           1       0.89      0.77      0.82       128
           2       0.72      0.83      0.77       100
           3       0.85      0.88      0.87        77

   avg / total       0.83      0.82      0.82       305
```

From figure 9 we can see that we got an accuracy of 81.6% from the decision tree classifier.

3.3.3 Random Forest classifier

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

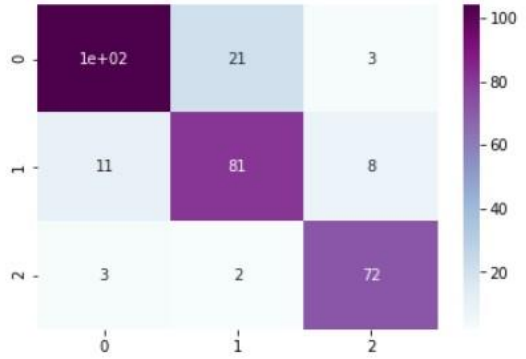
Figure 10- Training Random Forest classifier model

```
from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier()
model3 = rfc.fit(x_train, y_train)
prediction3 = model3.predict(x_test)
```

Figure 11- evaluating the model

```
In [134]: sns.heatmap(cm3, annot=True,cmap='BuPu')
Out[134]: <matplotlib.axes._subplots.AxesSubplot at 0x25bcd0d4128>
```




```
In [135]: accuracy_score(y_test, prediction3)
Out[135]: 0.8426229508196721

In [136]: print(classification_report(y_test, prediction3))
```

	precision	recall	f1-score	support
1	0.88	0.81	0.85	128
2	0.78	0.81	0.79	100
3	0.87	0.94	0.90	77
avg / total	0.84	0.84	0.84	305

From figure 11 we can see that we got an accuracy of 84.2% from the random forest classifier.

3.3.4 Support Vector Classification

Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features that we have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes well.

Figure 12- Training Support Vector Classification model

```
from sklearn.svm import SVC

svm=SVC()
model4=svm.fit(x_train,y_train)
prediction4=model4.predict(x_test)
cm4= confusion_matrix(y_test,prediction4)
```

Figure 13- evaluating the model

```
In [140]: sns.heatmap(cm4, annot=True,cmap='BuPu')
Out[140]: <matplotlib.axes._subplots.AxesSubplot at 0x25bcd15f400>
```


```
In [141]: accuracy_score(y_test, prediction4)
Out[141]: 0.6786885245901639

In [142]: print(classification_report(y_test, prediction4))
```

	precision	recall	f1-score	support
1	0.67	0.86	0.76	128
2	0.56	0.40	0.47	100
3	0.80	0.74	0.77	77
avg / total	0.67	0.68	0.67	305

From figure 13 we can see that we got an accuracy of 67.8% from support vector classification.

3.3.5 Gaussian Naive Bayes

Naive Bayes is a machine learning model that is used for large volumes of data, even if we are working with data that has millions of data records the recommended approach is Naive Bayes. It gives very good results when it comes to NLP tasks such as sentimental analysis. Gaussian Naive Bayes is a variant of Naive Bayes that follows Gaussian normal distribution and supports continuous data.

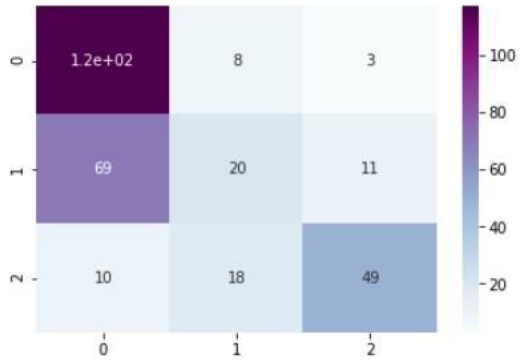
Figure 14- Training Gaussian Naive Bayes model

```
from sklearn.naive_bayes import GaussianNB

NB = GaussianNB()
model5 = NB.fit(x_train, y_train)
prediction5 = model5.predict(x_test)
cm5= confusion_matrix(y_test, prediction5)
```

Figure 15- training the model

```
In [145]: sns.heatmap(cm5, annot=True,cmap='BuPu')
Out[145]: <matplotlib.axes._subplots.AxesSubplot at 0x25bcd1e2748>
```




```
In [146]: accuracy_score(y_test, prediction5)
Out[146]: 0.6098360655737705

In [147]: print(classification_report(y_test, prediction5))
```

	precision	recall	f1-score	support
1	0.60	0.91	0.72	128
2	0.43	0.20	0.27	100
3	0.78	0.64	0.70	77
avg / total	0.59	0.61	0.57	305

From figure 15 we can see that we got an accuracy of 60.9% from gaussian naive bayes model.

3.3.6 K Nearest Neighbors

K-Nearest Neighbor is a ML algorithm based on Supervised Learning technique. It can be used for Regression as well as for Classification. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is similar to the available categories. It stores all the available data and classifies a new data point based on the similarity.

Figure 16- Training K Nearest Neighbors model

```
from sklearn.neighbors import KNeighborsClassifier

KNN = KNeighborsClassifier()
model6 = KNN.fit(x_train, y_train)
prediction6 = model6.predict(x_test)
cm6= confusion_matrix(y_test, prediction5)
```

Figure 17- Evaluating the model

```
In [149]: sns.heatmap(cm6, annot=True,cmap='BuPu')
Out[149]: <matplotlib.axes._subplots.AxesSubplot at 0x25bcd2759e8>
```


```
In [150]: accuracy_score(y_test, prediction6)
Out[150]: 0.6918032786885245

In [151]: print(classification_report(y_test, prediction6))
```

	precision	recall	f1-score	support
1	0.71	0.80	0.75	128
2	0.58	0.56	0.57	100
3	0.83	0.68	0.74	77
avg / total	0.70	0.69	0.69	305

From figure 17 we can see that the K nearest neighbor model gave an accuracy of 69.1%

3.4 Summary

Summarizing the results, we got from all the ML models

Figure 18- Summary of the accuracy of all the trained ML models

```
In [152]: print('lr Logistic Regression :', accuracy_score(y_test, prediction1))
          print('dtc Decision Tree Classifier :', accuracy_score(y_test, prediction2))
          print('rfc Random Forest Classifier :', accuracy_score(y_test, prediction3))
          print('NB: GaussianNB :', accuracy_score(y_test, prediction4))
          print('SVC Support Vector Classifier :', accuracy_score(y_test, prediction5))
          print('KNN K Nearest Neighbors :', accuracy_score(y_test, prediction6))

          lr Logistic Regression : 0.6524590163934426
          dtc Decision Tree Classifier : 0.8163934426229508
          rfc Random Forest Classifier : 0.8426229508196721
          NB: GaussianNB : 0.6786885245901639
          SVC Support Vector Classifier : 0.6098360655737705
          KNN K Nearest Neighbors : 0.6918032786885245
```

4. Conclusion

So, after comparing the results of all the models we can conclude that the Random Forest classifier gives us the best result (accuracy = 84.2%) and is the most suitable model for this dataset followed by the Decision tree classifier (accuracy = 81.6%).

From this project we can conclude that not all models are suited to a specific dataset we must test and find out which model would predict the target variable of the dataset more accurately and precisely and then use it as the final model for predicting the values.

6. References

- *Bertini, Ayleen et al. "Using Machine Learning to Predict Complications in Pregnancy: A Systematic Review." Frontiers in bioengineering and biotechnology vol. 9 780389. 19 Jan. 2022, doi:10.3389/fbioe.2021.780389*
- *Zeeshan Ahmed, Khalid Mohamed, Saman Zeeshan, XinQi Dong, Artificial intelligence with multi-functional machine learning platform development for better healthcare and precision medicine, Database, Volume 2020, 2020, baaa010, https://doi.org/10.1093/database/baaa010*
- *Boland, M.R., Polubriaginof, F. & Tatonetti, N.P. Development of A Machine Learning Algorithm to Classify Drugs Of Unknown Fetal Effect. Sci Rep 7, 12839 (2017). https://doi.org/10.1038/s41598-017-12943-x*
- *Davidson, L., Boland, M.R. Enabling pregnant women and their physicians to make informed medication decisions using artificial intelligence. J Pharmacokinet Pharmacodyn 47, 305–318 (2020). https://doi.org/10.1007/s10928-020-09685-1*

- *Emily F. Hamilton, Alina Dyachenko, Antonio Ciampi, Kimberly Maurel, Philip A. Warrick & Thomas J. Garite (2020) Estimating risk of severe neonatal morbidity in preterm births under 32 weeks of gestation, The Journal of Maternal-Fetal & Neonatal Medicine, 33:1, 73-80, DOI: 10.1080/14767058.2018.1487395*
- *https://archive.ics.uci.edu/ml/datasets/Maternal+Health+Risk+Data+Set#*
- *https://scikit-learn.org/stable/*