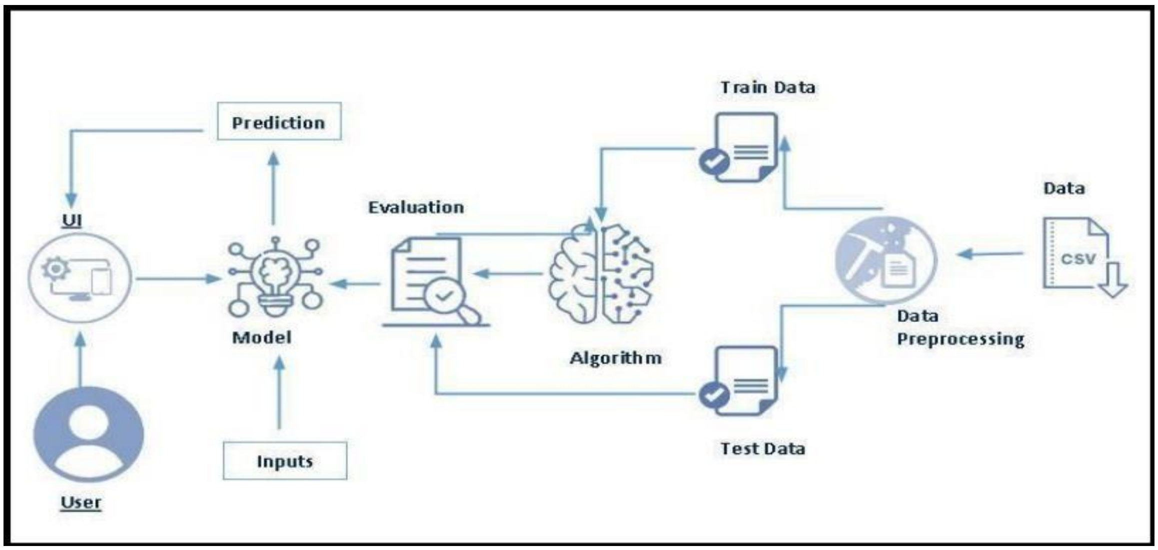
FetalAI: USING MACHINE LEARNING TO PREDICT AND MONITOR FETAL HEALTH

Reduction of child mortality is reflected in several of the United Nations' Sustainable Development Goals and is a key indicator of human progress. The UN expects that by 2030, countries end preventable deaths of newborns and children under 5 years of age, with all countries aiming to reduce under 5 mortalities to at least as low as 25 per 1,000 live births. Parallel to the notion of child mortality is of course maternal mortality, which accounts for 295 000 deaths during and following pregnancy and childbirth (as of 2017). The vast majority of these deaths (94%) occurred in low-resource settings, and most could have been prevented. In light of what was mentioned above, Cardiotocograms (CTGs) are a simple and cost accessible option to assess fetal health, allowing healthcare professionals to take action in order to prevent child and maternal mortality. The equipment itself works by sending ultrasound pulses and reading its response, thus shedding light on fetal heart rate (FHR), fetal movements, uterine contractions and more. In this project, we have some characteristics of Fetal Health as a dataset. The target variable of this dataset is Fetal Health. Since it is a multiclass classification, the classes are represented by ‘Normal’, ‘Pathological’ and ‘Suspect **Technical Architecture:**



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyzes the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below

- Define Problem / Problem Understanding
    - Specify the business problem
    - Business requirements
    - Literature Survey

  - Social or Business Impact.
- Data Collection & Preparation ○ Collect the dataset ○ Data Preparation
- Exploratory Data Analysis ○ Descriptive statistical ○ Visual Analysis
- Model Building ○ Training the model in multiple algorithms ○ Testing the model
- Performance Testing ○ Testing model with multiple evaluation metrics
- Model Deployment ○ Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation ○ Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

Project Structure:

Create the Project folder which contains files as shown below:

- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- model.pkl is our saved model. Further we will use this model for flask integration.
- Training folder contains a model training file.

Specify the business problem

Prefer the Project Description

Business requirements

The business requirement for fetal health classification typically arises in the healthcare industry, specifically in the obstetrics and gynecology (OB/GYN) department. The classification of fetal health is necessary to ensure the well-being of the unborn baby and to make informed decisions regarding pregnancy management.

In order to classify fetal health, healthcare providers typically use a variety of tools and techniques, including ultrasound, fetal monitoring, and other diagnostic tests. Machine learning and artificial intelligence algorithms can also be used to help classify fetal health based on various parameters such as heart rate, movement, and other physiological measures. These techniques can help healthcare providers to make more accurate and timely diagnosis and treatment decisions, leading to better health outcomes for both the mother and baby.

Literature Survey (Student Will Write)

A literature survey for a Fetal Health classification project would involve researching and reviewing existing studies, articles, and other publications on the topic of drug classification. The survey would aim to gather information on current classification systems, their strengths and weaknesses, and any gaps in knowledge that the project could address. The literature survey would also look at the methods and techniques used in previous classification projects, and any relevant data or findings that could inform the design and implementation of the current project.

Social or Business Impact.

Social Impact:

- Promoting Informed Decision-Making: - By providing accurate and up-to-date information on Fetal Health, can help expectant parents make informed decisions about their pregnancy and childbirth. For example, if a serious health issue is detected in the fetus, parents can decide whether to continue with the pregnancy or consider other options.
- Reducing Infant Mortality: - Access to information about fetal health can help expectant parents identify and treat potential health issues before they become life-threatening to the unborn child. This can help reduce the infant mortality rate and ensure that more babies are born healthy.

Business Model/Impact:

- Improved Patient Outcomes: By detecting potential health issues in fetuses early, healthcare providers can develop treatment plans that help ensure better outcomes for both the mother and the child. This can lead to improved patient satisfaction and retention rates.

- Increased Revenue: Healthcare providers who offer fetal health testing and monitoring services may be able to generate additional revenue streams from expectant parents who are willing to pay for these services. Additionally, if a health issue is detected in the fetus, additional tests, procedures, and treatments may be required, which can generate additional revenue for the healthcare provider.

• Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

• Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

• Importing the libraries

- Import the necessary libraries as shown in the image.

In [1]:

```
import numpy as np
import pandas as pd
pd.set_option('max_columns', None)

import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set_style('darkgrid')

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC, SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

import warnings
warnings.filterwarnings(action='ignore')
```

Read the Dataset

Our dataset format might be in .csv, excel files, txt, json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv () to read the dataset. As a parameter we have to give the directory of the csv file.

In [5]:

```
data = pd.read_csv("C:/Users/hp/Downloads/fetal_health.csv")
```

In [6]:

```
data.head()
```

Out[6]:

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations	abnormal_short_term_variab
0	120.0	0.000	0.0	0.000	0.000	0.0	0.0	
1	132.0	0.006	0.0	0.006	0.003	0.0	0.0	
2	133.0	0.003	0.0	0.008	0.003	0.0	0.0	
3	134.0	0.003	0.0	0.008	0.003	0.0	0.0	
4	132.0	0.007	0.0	0.008	0.000	0.0	0.0	

In [7]:

```
data.shape
```

Out[7]:

```
(2126, 22)
```

Data Preparation

As we have understood how the data is, let us pre-process the collected data.

The Machine Learning model cannot be trained on the imported data directly. The dataset might have randomness, we might have to clean the dataset and bring it in the right form. This activity involves the following steps:

- Handling Missing Values
- Handling Categorical Data
- Handling Imbalance Data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Handling Missing Values:

```
In [8]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2126 entries, 0 to 2125
Data columns (total 22 columns):
 #   Column                                                  Non-Null Count  Dtype
---  ------                                                  --------------  -----
 0   baseline value                                          2126 non-null   float64
 1   accelerations                                           2126 non-null   float64
 2   fetal_movement                                          2126 non-null   float64
 3   uterine_contractions                                    2126 non-null   float64
 4   light_decelerations                                     2126 non-null   float64
 5   severe_decelerations                                    2126 non-null   float64
 6   prolongued_decelerations                                2126 non-null   float64
 7   abnormal_short_term_variability                         2126 non-null   float64
 8   mean_value_of_short_term_variability                    2126 non-null   float64
 9   percentage_of_time_with_abnormal_long_term_variability  2126 non-null   float64
 10  mean_value_of_long_term_variability                     2126 non-null   float64
 11  histogram_width                                         2126 non-null   float64
 12  histogram_min                                           2126 non-null   float64
 13  histogram_max                                           2126 non-null   float64
```

```
1  data.isnull().sum()
```

```
baseline value                                            0
accelerations                                             0
fetal_movement                                            0
uterine_contractions                                      0
light_decelerations                                       0
severe_decelerations                                      0
prolongued_decelerations                                  0
abnormal_short_term_variability                           0
mean_value_of_short_term_variability                      0
percentage_of_time_with_abnormal_long_term_variability    0
mean_value_of_long_term_variability                       0
histogram_width                                           0
histogram_min                                             0
histogram_max                                             0
histogram_number_of_peaks                                 0
histogram_number_of_zeroes                                0
histogram_mode                                            0
histogram_mean                                            0
histogram_median                                          0
```

There are no missing values in the dataset. That is why we can skip this step.

Handling Categorical Data.

There are no categorical values in the dataset. That is why we can skip this step.

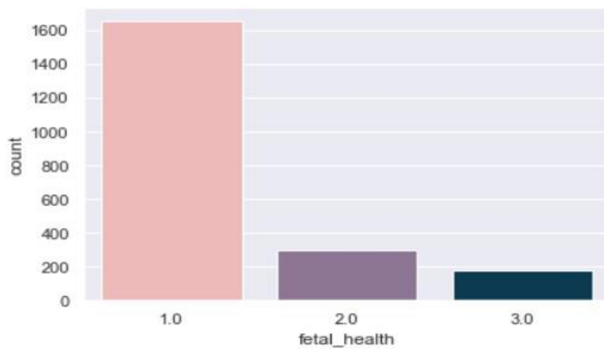
Handling Imbalance Data

```
In [139]: #first of all let us evaluate the target and find out if our data is imbalanced or not
          data['fetal_health'].value_counts()
```

```
Out[139]: 1.0    1655
          2.0     295
          3.0     176
          Name: fetal_health, dtype: int64
```

```
In [140]: colours=["#f7b2b0","#8f7198", "#003f5c"]
          sns.countplot(data= data, x="fetal_health",palette=colours)
```

```
Out[140]: <AxesSubplot:xlabel='fetal_health', ylabel='count'>
```



After checking, we get to know that the dataset is highly imbalanced. So, in the later stages we have balanced the dataset before training the model.

Exploratory Data Analysis

Descriptive statistics

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas have a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
In [66]: data.describe().T
```

Out[66]:

	count	mean	std	min	25%	50%	75%	max
baseline value	2126.0	133.303857	9.840844	106.0	126.000	133.000	140.000	160.000
accelerations	2126.0	0.003178	0.003866	0.0	0.000	0.002	0.006	0.019
fetal_movement	2126.0	0.009481	0.046666	0.0	0.000	0.000	0.003	0.481
uterine_contractions	2126.0	0.004366	0.002946	0.0	0.002	0.004	0.007	0.015
light_decelerations	2126.0	0.001889	0.002960	0.0	0.000	0.000	0.003	0.015
severe_decelerations	2126.0	0.000003	0.000057	0.0	0.000	0.000	0.000	0.001
prolongued_decelerations	2126.0	0.000159	0.000590	0.0	0.000	0.000	0.000	0.005
abnormal_short_term_variability	2126.0	46.990122	17.192814	12.0	32.000	49.000	61.000	87.000
mean_value_of_short_term_variability	2126.0	1.332785	0.883241	0.2	0.700	1.200	1.700	7.000
percentage_of_time_with_abnormal_long_term_variability	2126.0	9.846660	18.396880	0.0	0.000	0.000	11.000	91.000
mean_value_of_long_term_variability	2126.0	8.187629	5.628247	0.0	4.600	7.400	10.800	50.700

Visual analysis

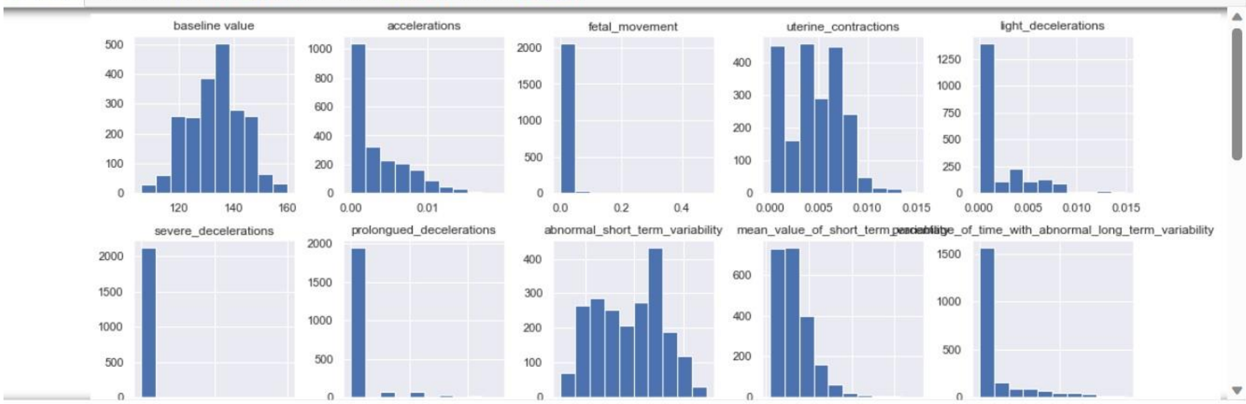
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Univariate analysis

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed different graphs such as histogram and boxplot.

The Seaborn and matplotlib package provide a wonderful functions histogram and boxplot. With the help of histogram and boxplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

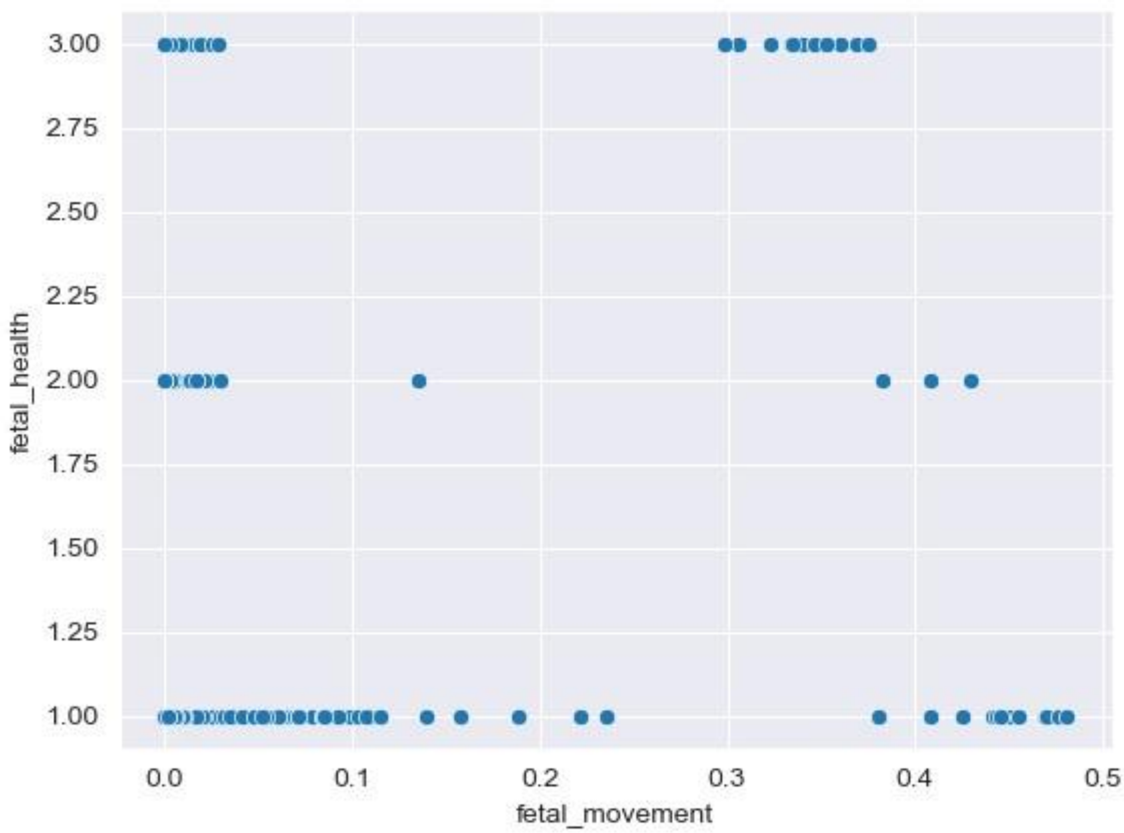
```
In [70]: data.hist(figsize=(17,17), layout=(5,5), sharex=False);
```



Bivariate Analysis:

```
sns.scatterplot(data.fetal_movement,data.fetal_health)
```

```
<AxesSubplot:xlabel='fetal_movement', ylabel='fetal_health'>
```



Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used correlation matrix.

```
In [129]: #correlation matrix
          corrmat= data.corr()
          plt.figure(figsize=(20,20))

          cmap = sns.light_palette("seagreen", as_cmap=True)

          sns.heatmap(corrmat,annot=True, cmap=cmap, center=0)
```

Feature Selection

```
In [130]: data.drop(columns=["histogram_mean"], axis=1, inplace=True)
```

```
In [131]: data.corr()["fetal_health"].sort_values(ascending=False)
```

```
Out[131]: fetal_health                                              1.000000
          prolongued_decelerations                                  0.484859
          abnormal_short_term_variability                           0.471191
          percentage_of_time_with_abnormal_long_term_variability    0.426146
          histogram_variance                                        0.206630
          baseline value                                            0.148151
          severe_decelerations                                      0.131934
          fetal_movement                                            0.088010
          histogram_min                                             0.063175
          light_decelerations                                       0.058870
          histogram_number_of_zeroes                               -0.016682
          histogram_number_of_peaks                                -0.023666
          histogram_max                                            -0.045265
          histogram_width                                          -0.068789
          mean_value_of_short_term_variability                     -0.103382
          histogram_tendency                                       -0.131976
          uterine_contractions                                     -0.204894
          histogram_median                                         -0.205033
          mean_value_of_long_term_variability                      -0.226797
          histogram_mode                                           -0.250412
          accelerations                                            -0.364066
          Name: fetal_health, dtype: float64
```

In [133]:
```
new_data=data.loc[:,["prolongued_decelerations", "abnormal_short_term_variability", "percentage_of_time_with_abnormal_long_term_v
```

In [134]:
```
new_data.head()
```

Out[134]:

	prolongued_decelerations	abnormal_short_term_variability	percentage_of_time_with_abnormal_long_term_variability	histogram_variance	histogram_median	me
0	0.0	73.0	43.0	73.0	121.0	
1	0.0	17.0	0.0	12.0	140.0	
2	0.0	16.0	0.0	13.0	138.0	
3	0.0	16.0	0.0	13.0	137.0	
4	0.0	16.0	0.0	11.0	138.0	

Scaling Data:

In [138]:
```
X = data.drop(columns=['fetal_health'])
y = data["fetal_health"]
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
X_scaled = pd.DataFrame(scale.fit_transform(X), columns=X.columns)
X_scaled.head()
```

Out[138]:

	accelerations	prolongued_decelerations	abnormal_short_term_variability	percentage_of_time_with_abnormal_long_term_variability	mean_value_of_long_term_va
0	0.000000	0.0	0.813333	0.472527	0.
1	0.315789	0.0	0.066667	0.000000	0.
2	0.157895	0.0	0.053333	0.000000	0.
3	0.157895	0.0	0.053333	0.000000	0.
4	0.368421	0.0	0.053333	0.000000	0.

Splitting data into train and test

In [141]:
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```

In [142]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 10)
X_train.shape, X_test.shape
```

Out[142]: ((1488, 8), (638, 8))

Applying SMOTE for balancing the data

```
In [146]: pip install imblearn

Requirement already satisfied: imblearn in c:\users\hp\anaconda3\lib\site-packages (
Requirement already satisfied: imbalanced-learn in c:\users\hp\anaconda3\lib\site-pa
Requirement already satisfied: joblib>=1.0.0 in c:\users\hp\anaconda3\lib\site-packa
0)
Requirement already satisfied: scikit-learn>=1.1.0 in c:\users\hp\anaconda3\lib\site
(1.1.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hp\anaconda3\lib\sit
n) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in c:\users\hp\anaconda3\lib\site-packa
2.4)
Requirement already satisfied: scipy>=1.3.2 in c:\users\hp\anaconda3\lib\site-packag
1)

[notice] A new release of pip available: 22.2 -> 23.1
[notice] To update, run: python.exe -m pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

```
In [147]: from imblearn.over_sampling import SMOTE
          smote = SMOTE()
```

```
In [148]: X_train_smote, y_train_smote = smote.fit_resample(X_train.astype('float'), y_train)
```

```
In [149]: from collections import Counter
          print ("Before SMOTE :" , Counter(y_train))
          print ("After SMOTE :" , Counter(y_train_smote))

Before SMOTE : Counter({1.0: 1158, 2.0: 201, 3.0: 129})
After SMOTE : Counter({1.0: 1158, 2.0: 1158, 3.0: 1158})
```

After applying SMOTE, the dataset is balanced. And now we will train the model after balancing the dataset to check the accuracy.

Model Building

Training the model in multiple algorithms

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying three classification algorithms. The best model is saved based on its performance.

Random Forest model

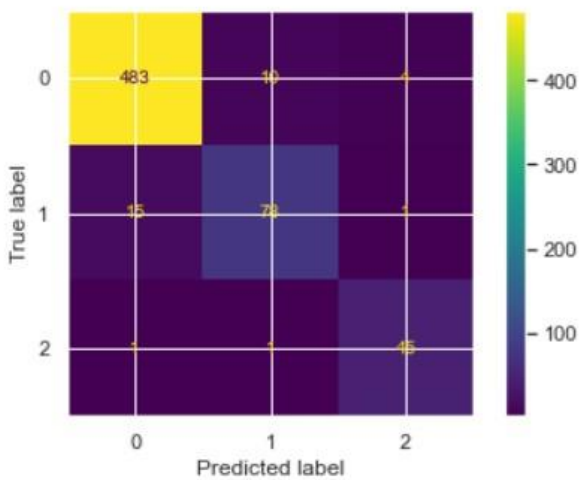
A function named randomForest is created and train and test data are passed as the parameters. Inside the function, the RandomForestClassifier algorithm is initialized and training data is passed to the model with the. fit () function. Test data is predicted with. predict () function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
In [153]: RF_model = RandomForestClassifier()
          RF_model.fit(X_train_smote, y_train_smote)
          predictions=RF_model.predict(X_test)
          print(accuracy_score(y_test,predictions))

          0.9498432601880877
```

```
In [155]: print("For the amounts of training data is: ",size)
          print("Accuracy of RandomForestClassifier: ",RF_model.score(X_test,y_test))
          cm = confusion_matrix(y_test, predictions)
          cm_display = ConfusionMatrixDisplay(cm).plot()
          plt.show()

          For the amounts of training data is:  3474
          Accuracy of RandomForestClassifier:  0.9498432601880877
```



Decision Tree

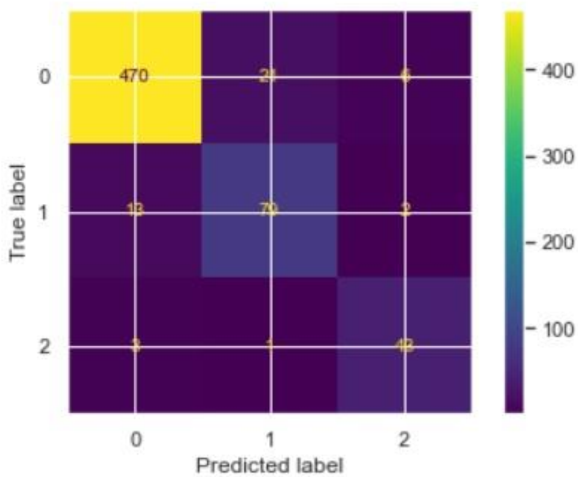
A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialized and training data is passed to the model with the. fit () function. Test data is predicted with. predict () function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
In [158]: DT_model = DecisionTreeClassifier()
          DT_model.fit(X_train_smote, y_train_smote)
          predictions = DT_model.predict(X_test)
          print(accuracy_score(y_test,predictions))

          0.9278996865203761
```

```
In [159]: print("For the amounts of training data is: ",size)
          print("Accuracy of DecisionTreeClassifier: ",DT_model.score(X_test,y_test))
          cm = confusion_matrix(y_test, predictions)
          cm_display = ConfusionMatrixDisplay(cm).plot()
          plt.show()

          For the amounts of training data is:  3474
          Accuracy of DecisionTreeClassifier:  0.9278996865203761
```



Logistic Regression

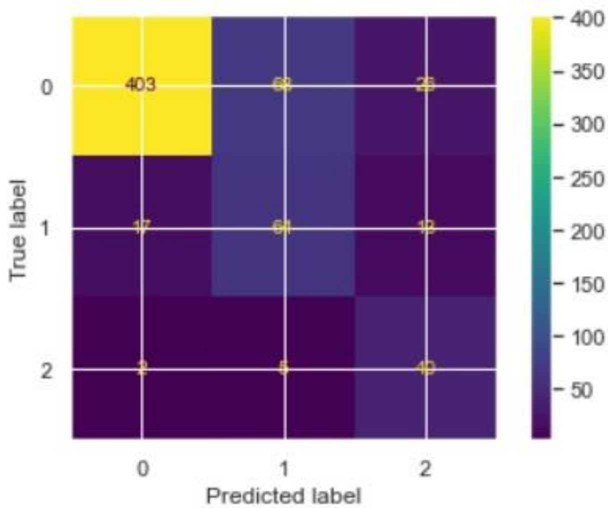
A function named LogisticRegression () is created and train and test data are passed as the parameters. Inside the function, LogisticRegression algorithm is initialized and training data is passed to the model with the. fit () function. Test data is predicted with. predict () function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
In [156]: LR_model = LogisticRegression()
          LR_model.fit(X_train_smote, y_train_smote)
          predictions = LR_model.predict(X_test)
          print(accuracy_score(y_test,predictions))

          0.7946708463949843
```

```
In [157]: print("For the amounts of training data is: ",size)
          print("Accuracy of LogisticRegression: ",LR_model.score(X_test,y_test))
          cm = confusion_matrix(y_test, predictions)
          cm_display = ConfusionMatrixDisplay(cm).plot()
          plt.show()
```

```
For the amounts of training data is:  3474
Accuracy of LogisticRegression:  0.7946708463949843
```



K-Nearest Neighbors

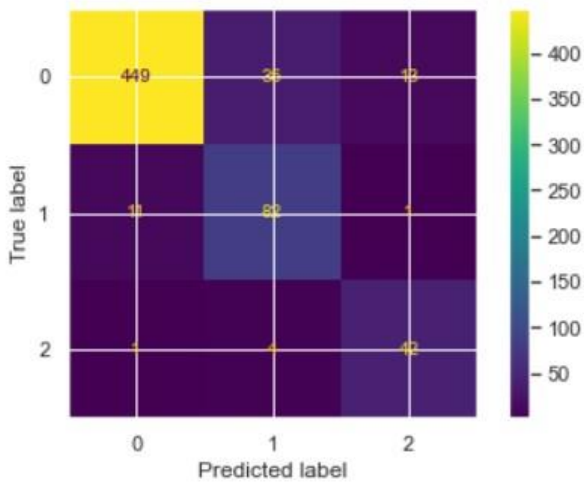
A function named KNeighborsClassifier () is created and train and test data are passed as the parameters. Inside the function, KNeighbors algorithm is initialized and training data is passed to the model with the. fit () function. Test data is predicted with. predict () function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

In [160]:
```python
KNN_model = KNeighborsClassifier(n_neighbors=5)
KNN_model.fit(X_train_smote, y_train_smote)
predictions = KNN_model.predict(X_test)
print(accuracy_score(y_test,predictions))
```

```
0.8981191222570533
```

In [161]:
```python
print("For the amounts of training data is: ",size)
print("Accuracy of KNeighborsClassifier: ",KNN_model.score(X_test,y_test))
cm = confusion_matrix(y_test, predictions)
cm_display = ConfusionMatrixDisplay(cm).plot()
plt.show()
```

```
For the amounts of training data is:  3474
Accuracy of KNeighborsClassifier:  0.8981191222570533
```



Testing the model

In [169]:
```python
RF_model.predict([[0.345, 0.1225, 23346, 0.23456, 0.987, 2345, 123, 0]])
```

Out[169]:
```
array([1.])
```

In [170]:
```python
RF_model.predict([[0.000, 0.0, 73.0, 43.0, 2.4, 73.0, 120.0, 121.0]])
```

Out[170]:
```
array([2.])
```

Performance Testing & Hyperparameter Tuning

Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

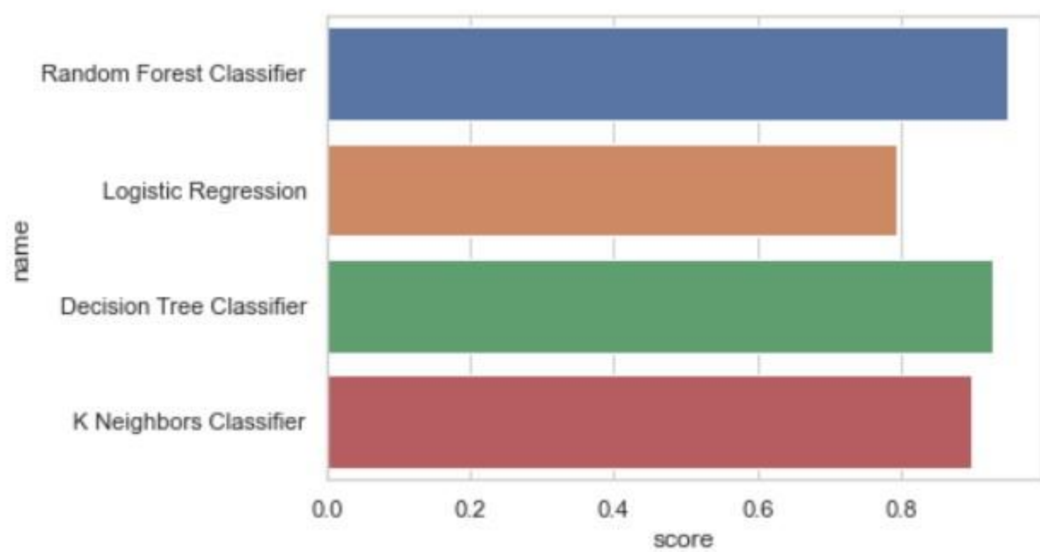
Comparing the model

```
In [165]: df = pd.DataFrame()
          df['name'] = names
          df['score'] = scores
          df
```

Out[165]:

	name	score
0	Random Forest Classifier	0.948276
1	Logistic Regression	0.794671
2	Decision Tree Classifier	0.929467
3	K Neighbors Classifier	0.898119

```
In [167]: sns.set(style="whitegrid")
          ax=sns.barplot(y="name", x="score", data=df)
```



After comparing the model with the help of bar plot. We came to a conclusion that Random Forest is showing the highest accuracy and is performing well.

Model Deployment

Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure

Create document as per the template provided