1 INTRODUCTION

DISCLAIMERS:

In no way does this project advocate for:

1) In no way does this project advocate for using this model INSTEAD of a clinical diagnosis. This model is designed to be used to empower birthing parents with more vital information sooner in their pregnancy.

2) In no way does this project advocate for what birthing parents should do, once they have their diagnosis.

WHAT THIS PROJECT DOES ADVOCATE FOR:

1) The choice of the birthing parent, empowered with the best information possible

2) Informed and empowered birthing parents

1.0.1 Problem: Maternal Morbidity on the Rise

According to the University of Minnesota, "Maternal deaths in the first months of the COVID-19 pandemic increased 33%—and even higher in Black and Hispanic women..."

CITATION: https://www.cidrap.umn.edu/news-perspective/2022/06/maternal-deaths-climbed-33-during-covid-19 (https://www.cidrap.umn.edu/news-perspective/2022/06/maternal-deaths-climbed-33-during-covid-19).

Duke University recently published a paper estimating that a total ban on abortion would increase Maternal morbidity rates dramatically.

"...I find that in the first year of such a ban, estimated pregnancy-related deaths would increase from 675 to 724 (49 additional deaths, representing a 7% increase), and in subsequent years to 815 (140 additional deaths, for a 21% increase). Non-Hispanic Black people would experience the greatest increase in deaths (a 33% increase in subsequent years). Estimated pregnancy-related deaths would increase for all races and ethnicities examined."

CITATION https://read.dukeupress.edu/demography/article/58/6/2019/265968/The-Pregnancy-Related-Mortality-Impact-of-a-Total (https://read.dukeupress.edu/demography/article/58/6/2019/265968/The-Pregnancy-Related-Mortality-Impact-of-a-Total)

Given the most recent overturning of Roe vs. Wade in the United States, it's crucial that medical research is performed with the goal of decreasing maternal morbidity. It's especially important that health care solutions for birthing parents be accessible and low cost.

1.0.2 Hypothetical Business Case: Maternal Morbidity Research

NYU Langone in NYC is doing research on Machine Learning and it's applications in lethal fetal diagnoses. Early detection in these cases provide the best possible medical outcomes for birthing parents and provide the most choices on how the birthing parents would like to handle the diagnosis.

1.0.3 Proposed Solution: An Early Detection AI for Pathological Diagnoses

I've been hired by NYU Langone to do design a model. I've decided that this model will focus on having a high recall as it's primary metric. This is because the risk of a potentially lethal pregnancy going undetected would result in 1 and/or two deaths, and the risk of a false positive would result in further medical diagnostics. This model is designed to flag pathological diagnoses as early as possible, for a doctor to confirm the diagnosis. This model is NOT designed to replace the diagnostic capabilities of a doctor, but to save time and resources for birthing parents and hospitals.

I'll be working with Cardiotocogram information specifically, and provide NYU Langone with my final result as proof of concept for further research.

1.0.4 Data Understanding

I'll be using Cardiotocogram information from CITATION:
https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification (https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification).

The original data is here: https://archive.ics.uci.edu/ml/datasets/cardiotocography (https://archive.ics.uci.edu/ml/datasets/cardiotocography)

This data was published in 2000, by Marques de SÃ¡, J.P., Biomedical Engineering Institute, Porto, Portugal. Bernardes, J., Faculty of Medicine, University of Porto, Portugal. Ayres de Campos, D., Faculty of Medicine, University of Porto, Portugal.

This is an appropriate data source for this problem/solution because it has a mix of Cardiotocogram information that range from normal to suspected diagnosis to pathological diagnosis.

LIMITATIONS: Pathological diagnosis in this data does not necessarily mean a lethal prenatal diagnosis, but that is the worst possible case that this data can represent. Therefore I'm going to be focusing on the "Pathological" class within this data, especially, and bear in mind that the more generalized and broad use cases for this model will be early detection of all pathological diagnoses, including lethal prenatal diagnosis.

My stakeholder: consider the following three stakeholders--

Stacey has a pre-existing medical condition putting her at high risk during her pregnancy. Where she lives, there are restrictions set in place, preventing her from getting an abortion after the first trimester. She wants to know as soon as possible if there is reason to suspect she has a lethal prenatal diagnosis, so that she can avoid carrying to term. Early diagnosis of fetal prenatel conditions will give her the time she needs, and prevent mental and physical trauma.

Alex has strong religious beliefs and knows she will want to carry her pregnancy to term and meet her child face to face if at all possible, even if that child has a fetal prenatal diagnosis. She wants to know as soon as possible if there is reason to suspect she has a lethal prenatal diagnosis, so that she can make funeral arrangements. Early diagnosis of lethal prenatel conditions will give her the time she needs and prevent mental and physical trauma.

NYU Langone is looking to provide the best possible outcomes to their patients, no matter how they decide to proceed with their respective diagnoses. Clinicians time needs to be prioritized wisely. Early interventions always provide less invasive and less risky outcomes. NYU Langone is looking to save resources and better allocate those resources by implementing early detection tools for folks like Stacey and Alex.

CITATION: https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification (https://www.kaggle.com/datasets/andrewmvd/fetal-health-classification)

CITATION : Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml] (http://archive.ics.uci.edu/ml%5D). Irvine, CA: University of California, School of Information and Computer Science.

From Kaggle on the Size of the Data and Classes within:

This dataset contains 2126 records of features extracted from Cardiotocogram exams, which were then classified by three expert obstetritians into 3 classes:

Normal

Suspect

Pathological

From Kaggle on the more general use cases of this data:

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 mortality to at least as low as 25 per 1,000 live births.

Parallel to 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.

Data This dataset contains 2126 records of features extracted from Cardiotocogram exams, which were then classified by three expert obstetricians into 3 classes:

- Normal

- Suspect
- Pathological

Further Reading: https://onlinelibrary.wiley.com/doi/10.1002/1520-6661(200009/10)9:5%3C311::AID-MFM12%3E3.0.CO;2-9 (https://onlinelibrary.wiley.com/doi/10.1002/1520-6661(200009/10)9:5%3C311::AID-MFM12%3E3.0.CO;2-9)

This is a paper by the original collectors of the data about their Machine Learning results.

1.0.5 Attribute Information:

LB - FHR baseline (beats per minute)

AC - # of accelerations per second

FM - # of fetal movements per second

UC - # of uterine contractions per second

DL - # of light decelerations per second

DS - # of severe decelerations per second

DP - # of prolongued decelerations per second

ASTV - percentage of time with abnormal short term variability

MSTV - mean value of short term variability

ALTV - percentage of time with abnormal long term variability

MLTV - mean value of long term variability

Width - width of FHR histogram

Min - minimum of FHR histogram

Max - Maximum of FHR histogram

Nmax - # of histogram peaks

Nzeros - # of histogram zeros

Mode - histogram mode

Mean - histogram mean

Median - histogram median

Variance - histogram variance

Tendency - histogram tendency

I'll be using all features in the initial modeling of the data, and I advocate for a secondary model with dimensionality reduction using SHAP with improved feature selection in the future.

1.0.6 IMPORTS

In [2]:

```
# Importing the nessisary libraries

import pandas as pd
import numpy as np
from itertools import cycle
import shap
import pickle
from IPython.display import Image

from sklearn import svm
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import label_binarize, LabelBinarizer

from sklearn.model_selection import GridSearchCV
from sklearn import metrics
from sklearn.metrics import recall_score, precision_score, accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_validate
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve, auc
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import roc_auc_score, plot_confusion_matrix, confusion_matrix
```

```
from sklearn import set_config
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

sns.set_style('white')
```

executed in 1.39s, finished 22:43:49 2022-07-05

1.0.7 FUNCTIONS

citation: https://stackoverflow.com/questions/68402691/adding-dropping-column-instance-into-a-pipeline (https://stackoverflow.com/questions/68402691/adding-dropping-column-instance-into-a-pipeline)

In [3]:

```
# creating a column transformer that only drops columns from a pipeline
```

executed in 1ms, finished 22:43:49 2022-07-05

In [4]:

```
class columnDropperTransformer():
    def __init__(self,columns):
        self.columns=columns

    def transform(self,X,y=None):
        return X.drop(self.columns,axis=1)

    def fit(self, X, y=None):
        return self
```

executed in 2ms, finished 22:43:49 2022-07-05

CITATION: https://stackoverflow.com/questions/46953967/multilabel-indicator-is-not-supported-for-confusion-matrix (https://stackoverflow.com/questions/46953967/multilabel-indicator-is-not-supported-for-

confusion-matrix)

In [5]:
```
# Creating an empty list called reports, and a function that will take a model,
# a y value and a set of predictions and evaluate how well the model's recall is
```

executed in 4ms, finished 22:43:49 2022-07-05

In [6]:
```
reports = []
```

executed in 2ms, finished 22:43:49 2022-07-05

In [7]:

```
def evaluate(model, y_val, y_preds):
    """
    DOCSTRING:
    evaluate expects a model, a list of y_trues and associated list of
    y_predicted. it outputs a confusion matrix of RECALL values (noramalize is
    set to true) and the associated precision, recall, f1 and support of the
    PATHOLOGICAL class (raw_report[2])
    """

    lables = ['Normal', 'Suspected', 'Pathological']

    ax = plt.subplot()

    cm = confusion_matrix(np.asarray(y_val).argmax(axis=1), np.asarray(y_preds).argmax(axis=1),
                          normalize = "true")

    sns.heatmap(cm, annot=True, fmt='.2%', ax=ax);

    #annot=True to annotate cells, ftm='g' to disable scientific notation

    # labels, title and ticks
    ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
    ax.set_title('Confusion Matrix');
    ax.xaxis.set_ticklabels(lables); ax.yaxis.set_ticklabels(lables);\

    raw_report = metrics.classification_report(y_val, y_preds, output_dict = True)
    class_3_report = raw_report['2']

    reports.append(class_3_report)

    print('************************************************************')
    print(cm)
```

```
print('METRICS FOR PATHOLOGICAL CLASS')
print(class_3_report)
```

executed in 4ms, finished 22:43:49 2022-07-05

CITATION: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html (https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html)

In [8]:

```
# creating a function that takes a list of predictions, a list of true y values,
# and the name of the model and provides an ROC AUC plot with multiclass labels
```

executed in 2ms, finished 22:43:49 2022-07-05

In [9]:

```
def visualize_ROC_AUC(y_preds, y_val, model_name):
    """
    DOCSTRING:
    visualize_ROC_AUC expects a list of predictions, a list of true y values,
    and the name of the model. It will output a multiclass ROC AUC plot using
    the OneVsRest method, making a line for each class, and two dotted lines,
    one for the micro_average (balenced for class imbalence), and macro_average
    (not balenced.)
    """

    # Compute ROC curve and ROC area for each class
    fpr = dict()
    tpr = dict()
    roc_auc = dict()
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_val[:, i], y_preds[:, i])
        roc_auc[i] = auc(fpr[i], tpr[i])

    # Compute micro-average ROC curve and ROC area
    fpr["micro"], tpr["micro"], _ = roc_curve(y_val.ravel(), y_preds.ravel())
    roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])


    # First aggregate all false positive rates
    all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))

    # Then interpolate all ROC curves at this points
    mean_tpr = np.zeros_like(all_fpr)
    for i in range(n_classes):
        mean_tpr += np.interp(all_fpr, fpr[i], tpr[i])

    # Finally average it and compute AUC
```

```
    mean_tpr /= n_classes

    fpr["macro"] = all_fpr
    tpr["macro"] = mean_tpr
    roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

    # Plot all ROC curves
    plt.figure()
    plt.plot(
        fpr["micro"],
        tpr["micro"],
        label="micro-average ROC curve (area = {0:0.2f}%)".format(roc_auc["micro"]),
        color="deeppink",
        linestyle=":",
        linewidth=4,
    )

    plt.plot(
        fpr["macro"],
        tpr["macro"],
        label="macro-average ROC curve (area = {0:0.2f}%)".format(roc_auc["macro"]),
        color="navy",
        linestyle=":",
        linewidth=4,
    )

    labels = {0 : "Normal", 1: "Suspect", 2: "Pathological"}


    colors = cycle(["aqua", "darkorange", "cornflowerblue"])
    for i, color in zip(range(n_classes), colors):
        plt.plot(
```

```
            fpr[i],
            tpr[i],
            color=color,
            lw=2,
            label=f"ROC curve of {labels[i]} (area = {round(roc_auc[i]*100)}%)"
        )

    plt.plot([0, 1], [0, 1], "k--", lw=2)
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title(f"{model_name}'s detection of different classes using 1 over all")
    plt.legend(loc="lower right")
    return plt.show()
```

executed in 7ms, finished 22:43:49 2022-07-05

In [10]:

```
# creating a function that will run evaluate and visualize together
```

executed in 1ms, finished 22:43:49 2022-07-05

In [11]:

```
def eval_and_vis(model, y_val, y_preds, model_name):
    evaluate(model, y_val, y_preds)
    visualize_ROC_AUC(y_preds, y_val, model_name)

```

executed in 2ms, finished 22:43:49 2022-07-05

In [12]:

```
# creating a function that will take model, X train, Y train and X test
# and will run evaluate and visualize on both the train and test sets
```

executed in 1ms, finished 22:43:49 2022-07-05

In [13]:

```
def show_train_and_test(grid, X_tr, y_tr, X_te, model_name):
    """
    DOCSTRING:
    show_train_and_test expects a grid or model, X_train, y_train, X_test
    and the name of the model. It outputs a report with the confusion matrix
    and ROC AUC plot and other metrics for both the test and train set
    """

    grid.fit(X_tr, y_tr)
    y_preds = grid.predict(X_tr)
    y_preds = lb.transform(y_preds)
    y_train_binarized = lb.transform(y_tr)
    print("TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT")
    eval_and_vis(grid, y_train_binarized, y_preds, 'model_name')


    y_preds = grid.predict(X_te)
    y_preds = lb.transform(y_preds)
    print("TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT")
    eval_and_vis(grid, y_val, y_preds, 'model_name')


    print("Best Parameters: \n{}\n".format(grid.best_params_))
    print('************************************************************')
```

executed in 3ms, finished 22:43:49 2022-07-05

In [14]:

```
# creating a function that takes a list of features and their impotances and
# plots them
```

executed in 1ms, finished 22:43:49 2022-07-05

1.0.8 Investigating the data

In [15]:
```
# loading the data
```
executed in 2ms, finished 22:43:49 2022-07-05

In [16]:
```
df = pd.read_csv('fetal_health.csv')
```
executed in 7ms, finished 22:43:49 2022-07-05

In [17]:
```
# taking a look at the first five rows
```
executed in 2ms, finished 22:43:49 2022-07-05

In [18]:
```
df.head()
```
executed in 15ms, finished 22:43:49 2022-07-05

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decelerations
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

5 rows × 22 columns

In [19]:
```
# looking at the size of the data
```
executed in 1ms, finished 22:43:49 2022-07-05

In [20]:
```
df.shape
```
executed in 3ms, finished 22:43:49 2022-07-05

```
(2126, 22)
```

In [21]:
```
# all columns are continuous with no apparent missing values
```
executed in 1ms, finished 22:43:49 2022-07-05

In [22]:
```
df.info()
```
executed in 7ms, finished 22:43:49 2022-07-05

```
<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
 14  histogram_number_of_peaks                               2126 non-null   float64
 15  histogram_number_of_zeroes                              2126 non-null   float64
 16  histogram_mode                                          2126 non-null   float64
 17  histogram_mean                                          2126 non-null   float64
 18  histogram_median                                        2126 non-null   float64
 19  histogram_variance                                      2126 non-null   float64
 20  histogram_tendency                                      2126 non-null   float64
 21  fetal_health                                            2126 non-null   float64
dtypes: float64(22)
memory usage: 365.5 KB
```

In [23]:
```
# no missing values are present
```
executed in 1ms, finished 22:43:49 2022-07-05

In [24]:
```
df.isna().sum()
```

executed in 4ms, finished 22:43:49 2022-07-05

```
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
histogram_variance                                            0
histogram_tendency                                            0
fetal_health                                                  0
dtype: int64
```

In [25]:
```
# taking a look at the discriptive statistics
```

executed in 1ms, finished 22:43:49 2022-07-05

In [26]:
```
1 df.describe()
```

executed in 37ms, finished 22:43:49 2022-07-05

	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued_decel
count	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000	2126.000000
mean	133.303857	0.003178	0.009481	0.004366	0.001889	0.000003	0.000159
std	9.840844	0.003866	0.046666	0.002946	0.002960	0.000057	0.000590
min	106.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	126.000000	0.000000	0.000000	0.002000	0.000000	0.000000	0.000000
50%	133.000000	0.002000	0.000000	0.004000	0.000000	0.000000	0.000000
75%	140.000000	0.006000	0.003000	0.007000	0.003000	0.000000	0.000000
max	160.000000	0.019000	0.481000	0.015000	0.015000	0.001000	0.005000

8 rows × 22 columns

Observations:

Fetal Movement doesn't have a value until it's upper quartile ranges, similar to light decelerations. Severe decelerations looks to be a very sparse feature, as well as prolonged decelerations.

In [27]:
```
1 # seperating the target from the rest of the data
```

executed in 2ms, finished 22:43:49 2022-07-05

In [28]:
```
1 X = df.drop(['fetal_health'],axis = 1)
2 y = df['fetal_health']
```

executed in 2ms, finished 22:43:49 2022-07-05

In [29]:
```
basic_labels = list(X.columns)
```
executed in 2ms, finished 22:43:49 2022-07-05

In [30]:
```
# Making a list of features for graphs
```
executed in 1ms, finished 22:43:49 2022-07-05

In [31]:
```
basic_labels = list(X.columns)
title_labels = [x.replace("_", " ").title() for x in basic_labels]
title_labels
```
executed in 2ms, finished 22:43:50 2022-07-05

```
['Baseline Value',
 'Accelerations',
 'Fetal Movement',
 'Uterine Contractions',
 'Light Decelerations',
 'Severe Decelerations',
 'Prolongued Decelerations',
 'Abnormal Short Term Variability',
 'Mean Value Of Short Term Variability',
 'Percentage Of Time With Abnormal Long Term Variability',
 'Mean Value Of Long Term Variability',
 'Histogram Width',
 'Histogram Min',
 'Histogram Max',
 'Histogram Number Of Peaks',
 'Histogram Number Of Zeroes',
 'Histogram Mode',
 'Histogram Mean',
 'Histogram Median',
 'Histogram Variance',
 'Histogram Tendency']
```

In [32]:
```
# making a dictionary to transform the column names of X
```
executed in 2ms, finished 22:43:50 2022-07-05

In [33]:
```
transform_labels = dict(zip(basic_labels, title_labels))
transform_labels
```

executed in 3ms, finished 22:43:50 2022-07-05

```
{'baseline value': 'Baseline Value',
 'accelerations': 'Accelerations',
 'fetal_movement': 'Fetal Movement',
 'uterine_contractions': 'Uterine Contractions',
 'light_decelerations': 'Light Decelerations',
 'severe_decelerations': 'Severe Decelerations',
 'prolongued_decelerations': 'Prolongued Decelerations',
 'abnormal_short_term_variability': 'Abnormal Short Term Variability',
 'mean_value_of_short_term_variability': 'Mean Value Of Short Term Variability',
 'percentage_of_time_with_abnormal_long_term_variability': 'Percentage Of Time With Abnormal Long Term Variability',
 'mean_value_of_long_term_variability': 'Mean Value Of Long Term Variability',
 'histogram_width': 'Histogram Width',
 'histogram_min': 'Histogram Min',
 'histogram_max': 'Histogram Max',
 'histogram_number_of_peaks': 'Histogram Number Of Peaks',
 'histogram_number_of_zeroes': 'Histogram Number Of Zeroes',
 'histogram_mode': 'Histogram Mode',
 'histogram_mean': 'Histogram Mean',
 'histogram_median': 'Histogram Median',
 'histogram_variance': 'Histogram Variance',
 'histogram_tendency': 'Histogram Tendency'}
```

In [34]:
```
# cleaning up X
```

executed in 2ms, finished 22:43:50 2022-07-05

In [35]:
```
X = X.rename(mapper = transform_labels, axis = 1)
```

executed in 2ms, finished 22:43:50 2022-07-05

In [36]:
```
# taking a look at the class imbalence and saving the ratio
```

executed in 1ms, finished 22:43:50 2022-07-05

In [37]:
```
y_distribution = y.value_counts()
y_distribution
```

executed in 3ms, finished 22:43:50 2022-07-05

```
1.0    1655
2.0     295
3.0     176
Name: fetal_health, dtype: int64
```

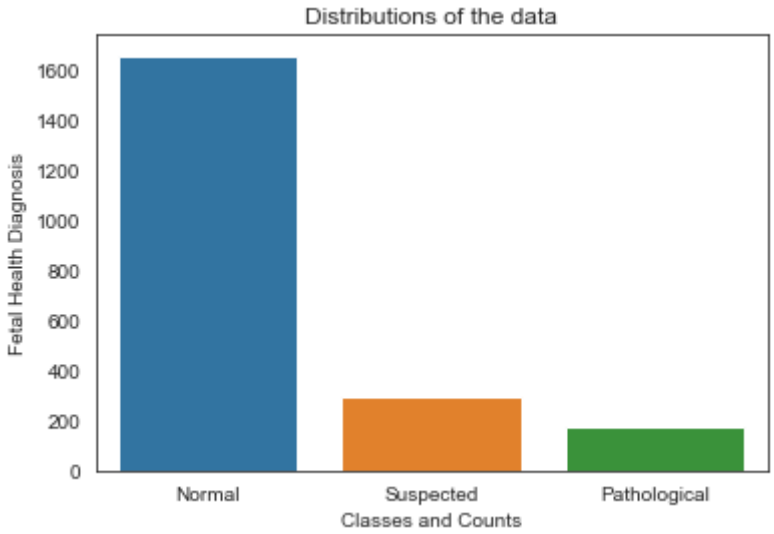
In [38]:
```
# visualizing the class imbalence
```

executed in 2ms, finished 22:43:50 2022-07-05

In [39]:

```
fig, ax = plt.subplots()
distributions = sns.barplot(x = y_distribution,
                            y = y_distribution.index,
                            data = y_distribution,
                           order = y_distribution)
distributions.set_title("Distributions of the data")
distributions.set_xlabel("Classes and Counts")
plt.xticks(ticks = [0, 1, 2], labels = ["Normal", "Suspected", "Pathological"])
plt.ylabel("Fetal Health Diagnosis");
```

executed in 87ms, finished 22:43:50 2022-07-05



In [40]:

```
# exploring the ratios of the class imbalence and saving those as weights
# to try as params on models
```

executed in 2ms, finished 22:43:50 2022-07-05

In [41]:
```
y_dict = dict(y.value_counts(normalize = True))
y_dict

y_weights = {}
for key, value in y_dict.items():
    key = key - 1
    y_weights[key] = 1 - value
y_weights
```

executed in 5ms, finished 22:43:50 2022-07-05

```
{0.0: 0.22154280338664156, 1.0: 0.861241768579492, 2.0: 0.9172154280338665}
```

Observations: there is a heavy class imbalance that must be accounted for in modeling.

In [42]:
```
# investigating the columns, aggrigated by the target
```

executed in 2ms, finished 22:43:50 2022-07-05

In [43]:
```
mean_agg = df.groupby(by = df['fetal_health']).mean()
mean_agg
```

executed in 12ms, finished 22:43:50 2022-07-05

fetal_health	baseline value	accelerations	fetal_movement	uterine_contractions	light_decelerations	severe_decelerations	prolongued
1.0	131.981873	0.003992	0.007963	0.004781	0.001941	6.042296e-07	0.000051
2.0	141.684746	0.000275	0.008332	0.002390	0.000536	0.000000e+00	0.000095
3.0	131.687500	0.000392	0.025676	0.003784	0.003670	3.409091e-05	0.001273

3 rows × 21 columns

1.0.9 Investigating the Attributes's Relationship with the Target

CITATION: https://www.kaggle.com/code/casper6290/fetalhealthclassification-99 (https://www.kaggle.com/code/casper6290/fetalhealthclassification-99)

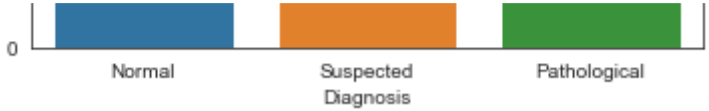
In [44]:
```
# making a plot to look at the classes by feature
```

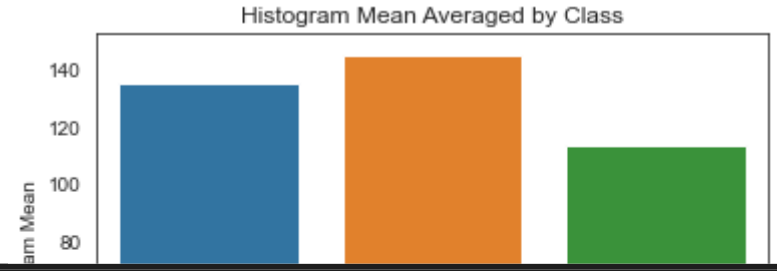
executed in 1ms, finished 22:43:50 2022-07-05

In [45]:

```
counter = 0
for index, col in enumerate(mean_agg.columns,start=1):
    fig, ax = plt.subplots()
    figure = sns.barplot(data = mean_agg,
                         x = mean_agg.index,
                         y = mean_agg[col])

    figure.set_title(f"{title_labels[counter]} Averaged by Class")
    figure.set_xlabel("Diagnosis")
    figure.set_ylabel(f"{title_labels[counter]}")
    plt.xticks(ticks = [0, 1, 2], labels = ["Normal", "Suspected", "Pathological"])
#    plt.savefig(f"{title_labels[counter]}")
    counter += 1
```

executed in 1.44s, finished 22:43:51 2022-07-05





Observations and Implications:

- Baseline Values

  is pretty equally distributed throughout all three classes.

- Accelerations

  were pretty equal between suspected and pathological, but ranged much much higher in the normal class. Further investigation upon this attribute is justified, as it may be important for ruling out the pathological diagnosis.

- Fetal Movement

  is roughly equal between normal and suspected classes but much higher with pathological cases, further investigation upon this attribute is justified, as it may be highly correlated with the pathological class.

- Uterine Contractions

  is an interesting attribute as it's highest value is in the normal class, but it's second highest value is in the pathological class, followed by the suspected class. This implies that the model may find this feature confusing for distinguishing normal from pathological classes.

- Light Decelerations

  is another interesting attribute as it's highest value is the pathological class, followed by the normal class at about half as high, followed by the suspected class at about 1/3rd as high as the normal class. This is a feature that could be fantastic at picking out a suspected diagnosis, but is of less interest to my pathological focused business case.

- Severe Decelerations

  looks to be an important attribute for selecting pathological diagnoses, as it's almost exclusively present in the pathological class, and almost not found in any other class.

- Prolonged Decelerations

  looks to be an important indicator for similar reasons, however it is somewhat more present equally between normal and suspected classes, it looks to be about eight times as high in pathological cases.

- Abnormal Short Term Variability

has implications of being somewhat useful in ruling out a case as normal, as it's about a third less high in normal cases as it is among suspected and pathological cases.

- Mean Value of Short Term Variability

  is an interesting attribute as it's highest in the pathological class, but nearly as high in the normal class, and about half as high in the suspected class. It would be useful in distinguishing the suspected class from the other two classes.

- Percentage of Time with Abnormal Short Term Variability

  is highest in the suspected class and about a 1/5th less high in the pathological class. Compared to the normal class, both the suspected and pathological classes are much higher in this attribute. Therefore this attribute is probably best at ruling out cases as normal for our purposes.

- Mean value of Long Term Variability

  is about halved in the pathological class as it is in the suspected and normal classes, making implying it may be useful in distinguishing pathological cases from the other two classes.

- Histogram Width

  is about equal between the pathological class and the normal class, but about half as high in the suspected class. This implies this attribute would be good at distinguishing suspected cases from the normal and pathological classes.

- Histogram Min

  is highest in suspected cases, and only slightly higher than it is in pathological and normal cases. This implies this feature may not be very useful to our model, but may give slight advantage if our goal was to pick a suspected case out between both pathological and normal cases.

- Histogram Max

  is about equal between all three classes, implying that this feature may not be very significant to any of the classes.

- Histogram Number of Peaks

is highest in the pathological class, but almost equal with the normal class. However, it's about 1 less on average from the pathological class, implying it may have some significants in ruling out suspected for our purposes.

- Histogram Number of Zeros

  follows a very similar pattern.

- Histogram Mode

  is about 20 less in the pathological class than it is in the other two classes, implying that it may be useful for our purposes of isolating pathological cases.

- Histogram Mean and Histogram Median

  follow a similar pattern but all three classes have a bit less distinction between them in these attributes.

- Histogram Variance

  more than three as high as the normal class, which is about three times as high as the suspected class, implying this attribute will be very useful for isolating pathological cases, and making each class distinct to the model.

- Histogram Tendency

  looks to be a very important attribute to our model, as in the pathological class, this value averages a negative value, as opposed to the two other classes where the values are positive and the absolute values are much higher. This attribute looks to be encoded by the data collectors, and I'd like to know much more about what these values mean to a non-technical audience. In my business case, I'd want to communicate with the data collection team about this attribute.

In [46]:
```
# splitting X train and X test data
```

executed in 2ms, finished 22:43:51 2022-07-05

In [47]:
```
X_train, X_holdout, y_train, y_holdout = train_test_split(X, y, random_state=42, test_size = .1)
```

executed in 3ms, finished 22:43:51 2022-07-05

In [48]:
```
# making a small holdout set
```
executed in 1ms, finished 22:43:51 2022-07-05

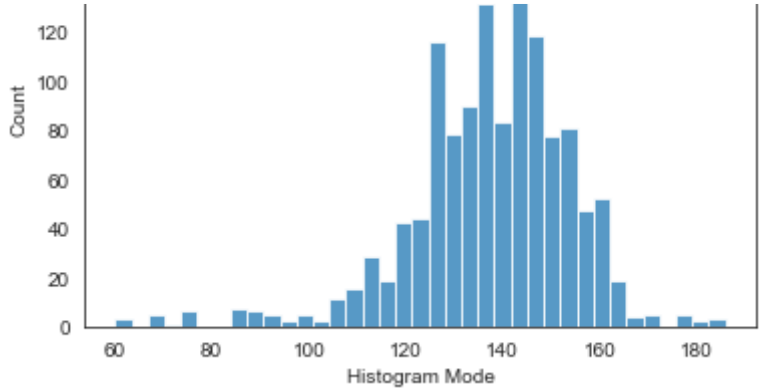
In [49]:
```
X_train,  X_val , y_train, y_val = train_test_split(X_train, y_train, test_size=0.33, random_state
```
executed in 3ms, finished 22:43:51 2022-07-05

In [50]:
```
# looking at the distributions of all features in X Train
```
executed in 2ms, finished 22:43:51 2022-07-05

In [51]:

```
counter = 0

for col in X_train:
    fig, ax = plt.subplots()
    distributions = sns.histplot(data = X_train, x = col)
    distributions.set_title(f'{title_labels[counter]} Distributions in the Data')
    distributions.set_xlabel(f'{title_labels[counter]}')
    counter += 1;
```

executed in 3.12s, finished 22:43:54 2022-07-05



In [52]:

```
# OHE y using Label Binarizer (this is so my ROC AUC plots will work)
```

executed in 1ms, finished 22:43:54 2022-07-05

In [53]:
```
lb = LabelBinarizer()

y_train_binarized = lb.fit_transform(y_train)
y_train_binarized

n_classes = 3
```

executed in 3ms, finished 22:43:54 2022-07-05

Correlation to the target class:

In [54]:
```
# making a df to plot the correlations to the target
```

executed in 5ms, finished 22:43:54 2022-07-05

In [55]:

```
y_train_target_pathological = [x[2] for x in y_train_binarized]

correlation_df = X_train.copy()
correlation_df["Pathological"] = y_train_target_pathological

correlation_df.head()
```

executed in 14ms, finished 22:43:54 2022-07-05

	Baseline Value	Accelerations	Fetal Movement	Uterine Contractions	Light Decelerations	Severe Decelerations	Prolongued Decelerations	Abnormal Short Term Variability	Mean Value Of Short Term Variability	Percen Of T Abno Long T Variat
125	159.0	0.000	0.0	0.003	0.0	0.0	0.000	65.0	0.4	16.0
922	122.0	0.002	0.0	0.002	0.0	0.0	0.001	31.0	1.2	1.0
1390	142.0	0.000	0.0	0.007	0.0	0.0	0.000	60.0	0.4	8.0
1446	147.0	0.003	0.0	0.004	0.0	0.0	0.000	46.0	0.7	46.0
1032	136.0	0.013	0.0	0.005	0.0	0.0	0.000	22.0	2.3	0.0

5 rows × 22 columns

In [56]:

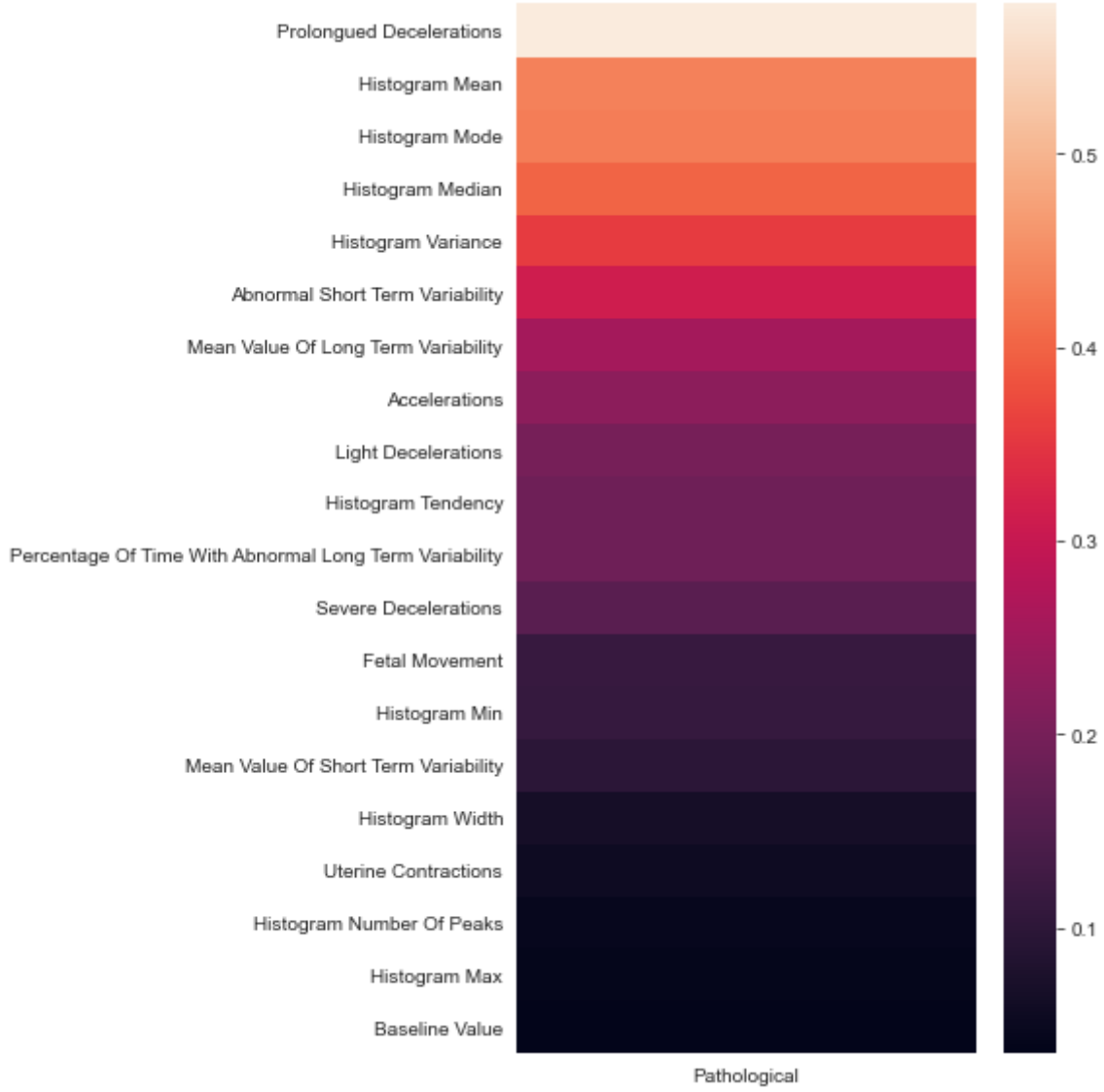
```
# making a heat map between the features of X train and the target variable
```

executed in 2ms, finished 22:43:54 2022-07-05

In [209]:

```
target_correlation = abs(correlation_df.corr())[['Pathological']].sort_values(by = "Pathological")
target_correlation = target_correlation[1:-1]

plt.figure(figsize = (8, 8))
target_correlation_plot = sns.heatmap(data = target_correlation)
target_correlation_plot.invert_yaxis()

plt.tight_layout()
#plt.savefig("Target Correlation")
```

executed in 425ms, finished 11:22:38 2022-07-06



In [58]:
```
# Plotting those correlations on a bar chat
```

executed in 2ms, finished 22:43:54 2022-07-05

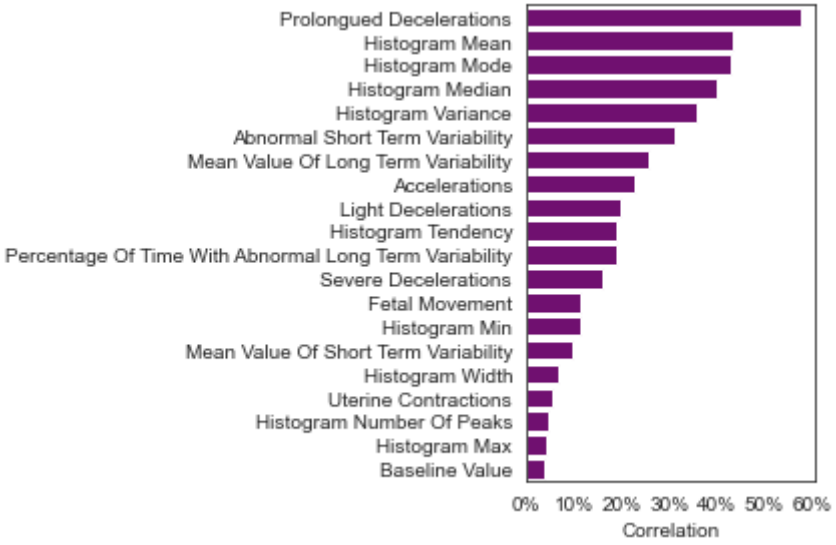
In [225]:
```
target_correlation.sort_values(by = "Pathological", ascending = False, inplace = True)
```

executed in 7ms, finished 11:35:52 2022-07-06

In [227]:

```
fig, ax = plt.subplots()

target_correlation_bar_plot = sns.barplot(data = target_correlation,
                                          x = target_correlation.values.flatten(),
                                          y = target_correlation.index,
                                         color = "purple")

target_correlation_bar_plot.set_xlabel("Correlation")
plt.xticks(ticks = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6], labels = ["0%", "10%",
                                                                "20%", "30%",
                                                                "40%", "50%",
                                                                "60%"]);
plt.tight_layout()
plt.savefig("Target Correlation Barplot");
```

executed in 224ms, finished 11:35:58 2022-07-06



In [60]:
```
# this is a good hypothosis for what the feature importances will be
```
executed in 1ms, finished 22:43:55 2022-07-05

1.1 Data Prep:

Due to this data being remarkably clean, (there are no missing or NAN values, I won't be doing much to the data in prep for modeling. In certain models I'll be scaling the data, but that will happen on a case by case basis as it's not always appropriate for each kind of model.

2 MODELING

setting my display to output a diagram

In [61]:
```
set_config(display = 'diagram')
```
executed in 1ms, finished 22:43:55 2022-07-05

transforming y_val for my confusion matrixes and ROC plots

In [62]:
```
y_val = lb.transform(y_val)
```
executed in 3ms, finished 22:43:55 2022-07-05

2.1 Logistic Regression

JUSTIFICATION: This is an easy to interpret model, and I will explore these coefficiants in future work for a best possible inferential model.

2.1.1 Logistic Regression Baseline

In [63]:

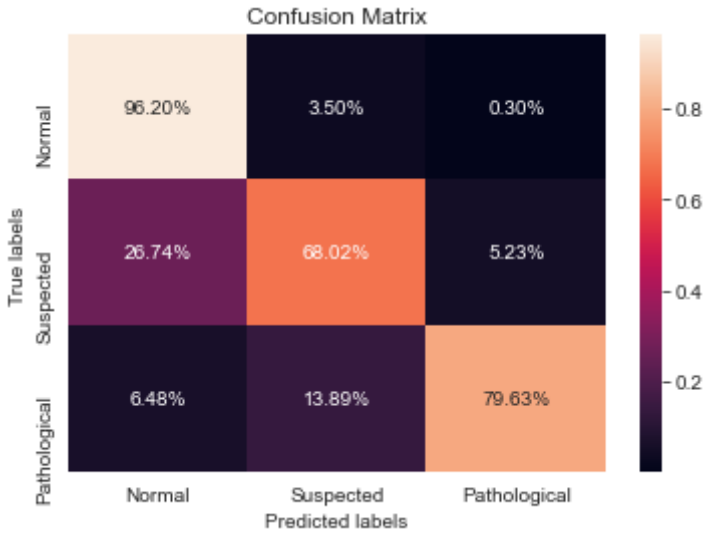
```
lg_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('lg', LogisticRegression()),

])
baseline_params = {'lg__random_state' : [42]}

grid = GridSearchCV(lg_pipeline, baseline_params, scoring = 'recall_micro')
```

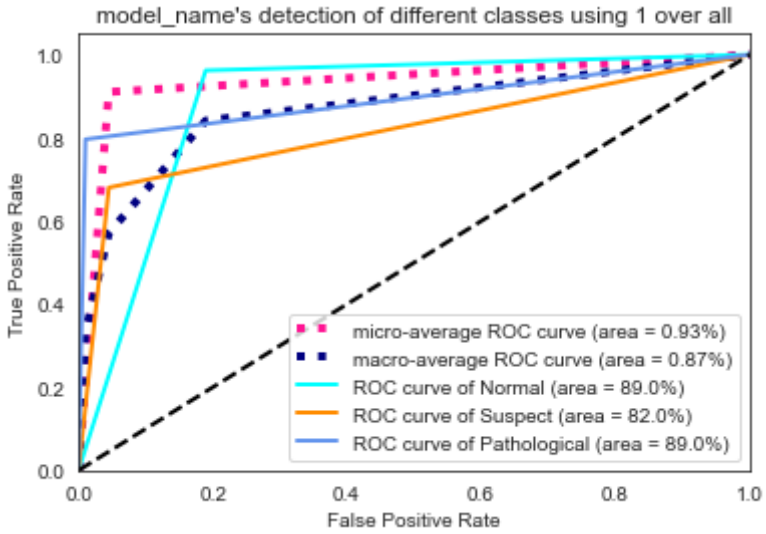
executed in 2ms, finished 22:43:55 2022-07-05

In [64]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'lg_pipeline')
```

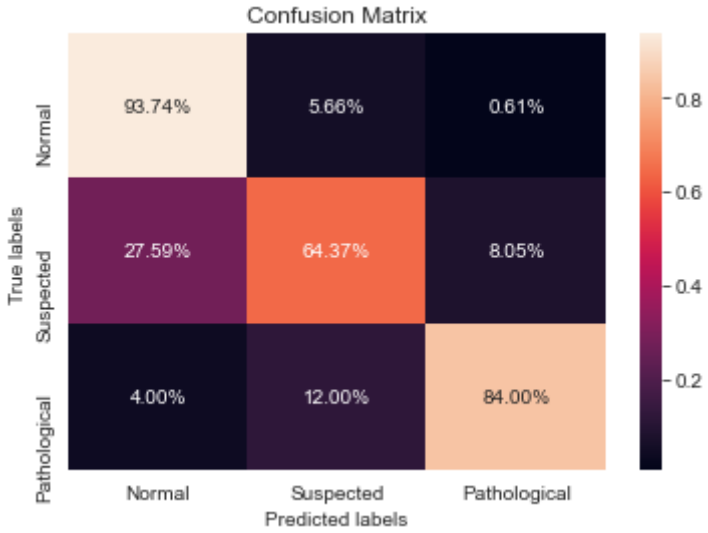
executed in 619ms, finished 22:43:55 2022-07-05

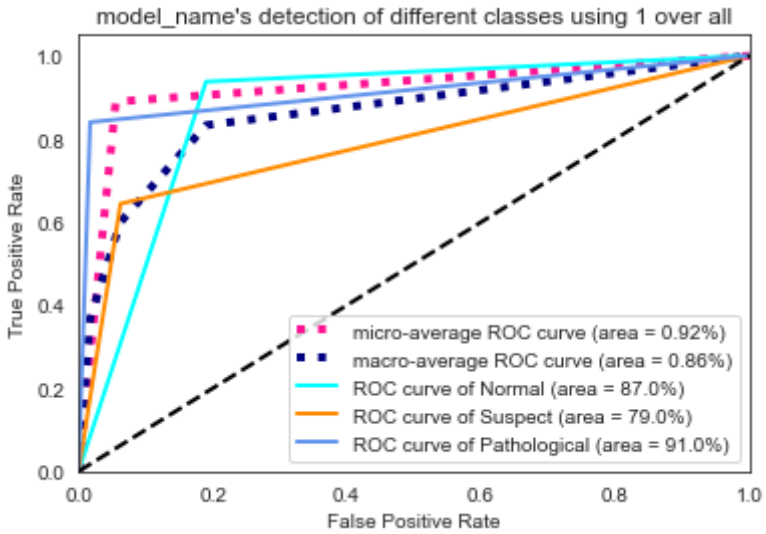
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.96203796 0.03496503 0.002997  ]
 [0.26744186 0.68023256 0.05232558]
 [0.06481481 0.13888889 0.7962963 ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8775510204081632, 'recall': 0.7962962962962963, 'f1-score': 0.8349514563106796, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.93737374 0.05656566 0.00606061]
 [0.27586207 0.64367816 0.08045977]
 [0.04       0.12       0.84      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8076923076923077, 'recall': 0.84, 'f1-score': 0.8235294117647058, 'support': 50}
```






```
Best Parameters:
{'lg__random_state': 42}

************************************************************
```

2.1.2 Logistic Regression Gridsearch

In [65]:
```
# now trying a gridsearch to tune the model further
```

executed in 1ms, finished 22:43:55 2022-07-05

In [66]:

```
lg_gscv_params = {
    'lg__C': [0.01, 1, 10, 100],
    'lg__penalty' : ['l1', 'l2', 'elasticnet'],
    'lg__solver' : ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga'],
    'lg__class_weight': [y_weights, 'balanced'],
    'lg__multi_class' : ['auto', 'ovr', 'multinomial']}
```

executed in 2ms, finished 22:43:55 2022-07-05

In [67]:

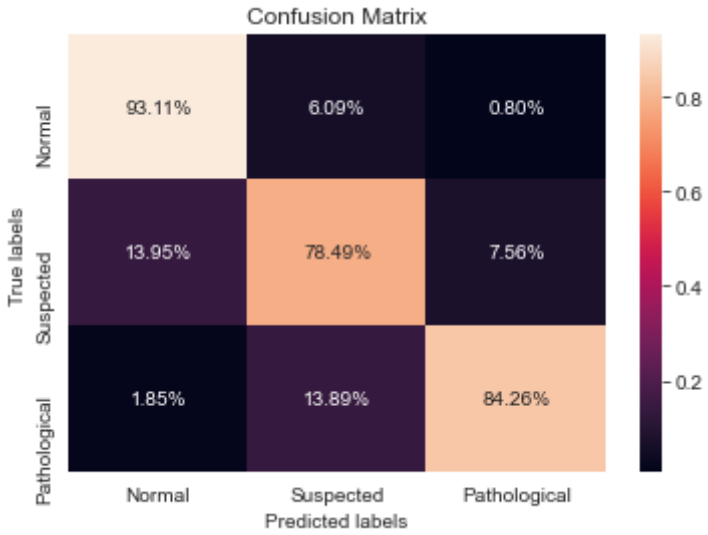
```
grid = GridSearchCV(lg_pipeline, lg_gscv_params, scoring = 'recall_micro')
```

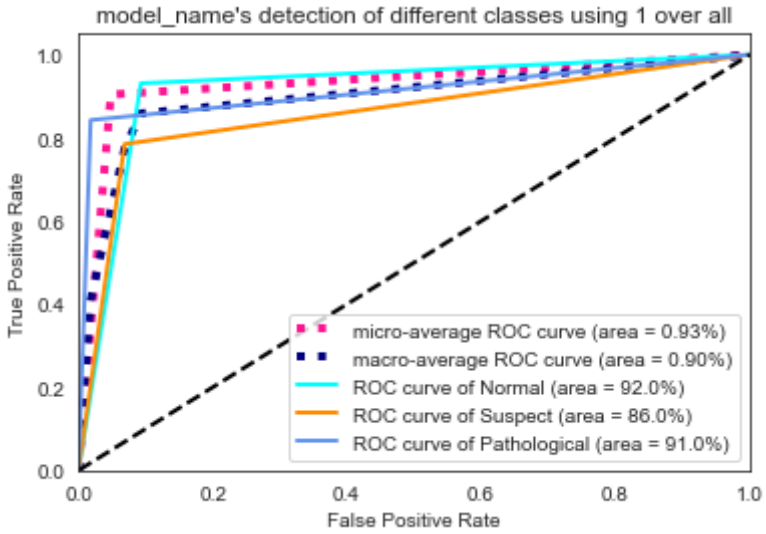
executed in 2ms, finished 22:43:55 2022-07-05

In [68]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'lg_gscv')
```

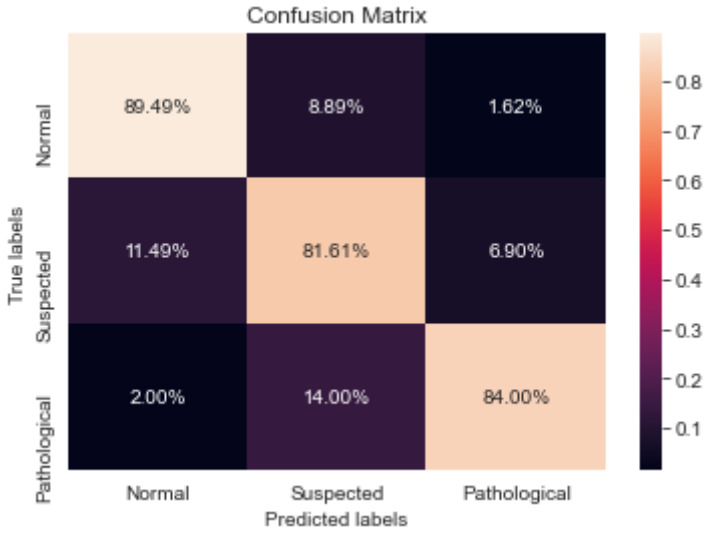
executed in 20.9s, finished 22:44:16 2022-07-05

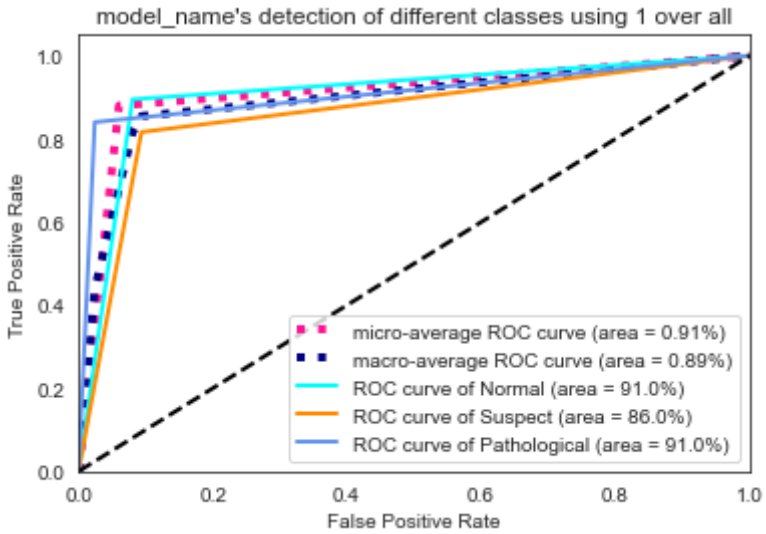
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.93106893 0.06093906 0.00799201]
 [0.13953488 0.78488372 0.0755814 ]
 [0.01851852 0.13888889 0.84259259]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8125, 'recall': 0.8425925925925926, 'f1-score': 0.8272727272727272, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.89494949 0.08888889 0.01616162]
 [0.11494253 0.81609195 0.06896552]
 [0.02       0.14       0.84      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.75, 'recall': 0.84, 'f1-score': 0.7924528301886793, 'support': 50}
```






```
Best Parameters:
{'lg__C': 1, 'lg__class_weight': 'balanced', 'lg__multi_class': 'auto', 'lg__penalty': 'l1', 'lg__solver': 'liblinear'}

**********************************************************
```

Observations:

This model doesn't seem to be overfit. It performs better on the Suspected class than the model before grid searching.

2.2 OneVsRest Logistic Regression

2.2.1 OneVsRest Logistic Regression Baseline

JUSTIFICATION: Since I'm more interested in one class of the target variable, I'm interested to see if using a OneVsRest model might give me an edge on predicting Pathological diagnoses over either Suspected or Normal.

Making a baseline OneOverAll Logistic Resgression Model

In [69]:

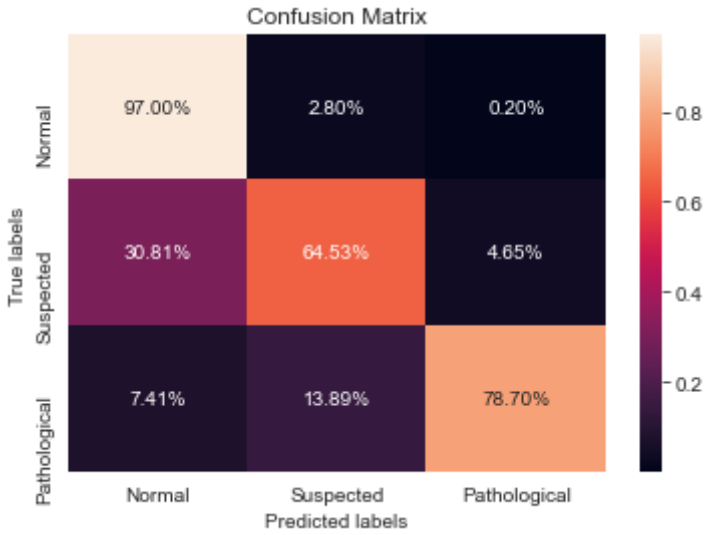
```
OVR_lg_pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('OVR_lg', OneVsRestClassifier(LogisticRegression())),

])
baseline_params = {}

grid = GridSearchCV(OVR_lg_pipeline, baseline_params, scoring = 'recall_micro')
```

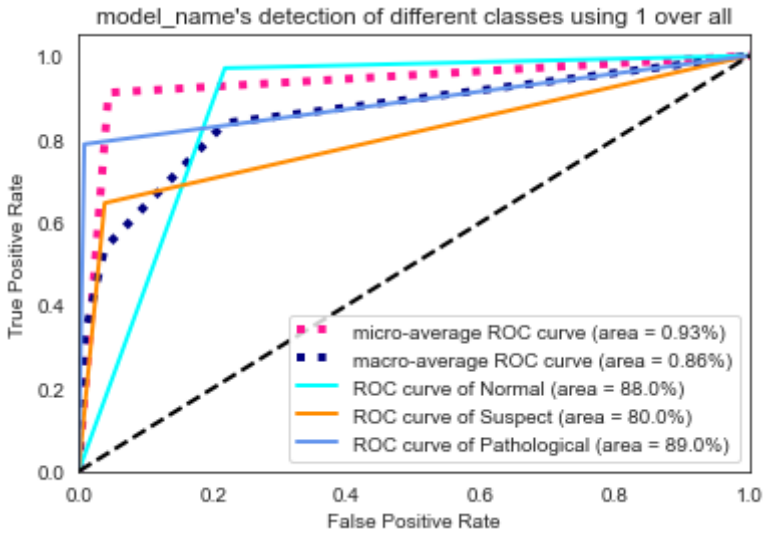
executed in 3ms, finished 22:44:16 2022-07-05

In [70]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'OVR_lg_pipeline')
```

executed in 604ms, finished 22:44:17 2022-07-05

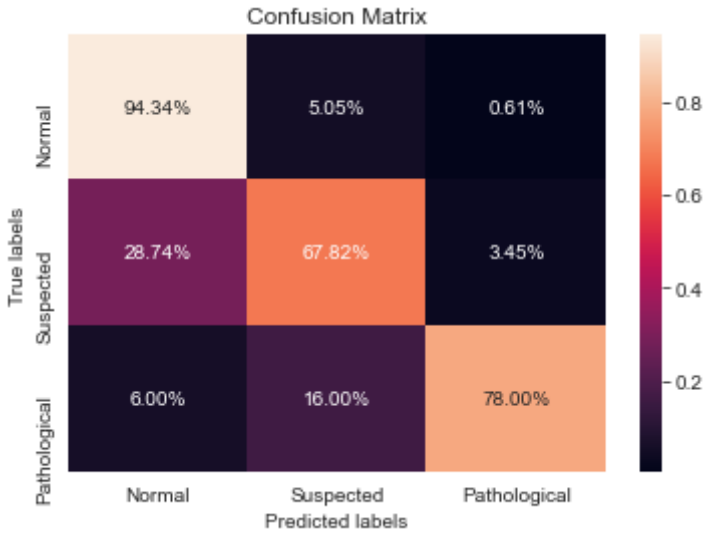
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.97002997 0.02797203 0.001998  ]
 [0.30813953 0.64534884 0.04651163]
 [0.07407407 0.13888889 0.78703704]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8947368421052632, 'recall': 0.7870370370370371, 'f1-score': 0.8374384236453202, 'support': 108}
```

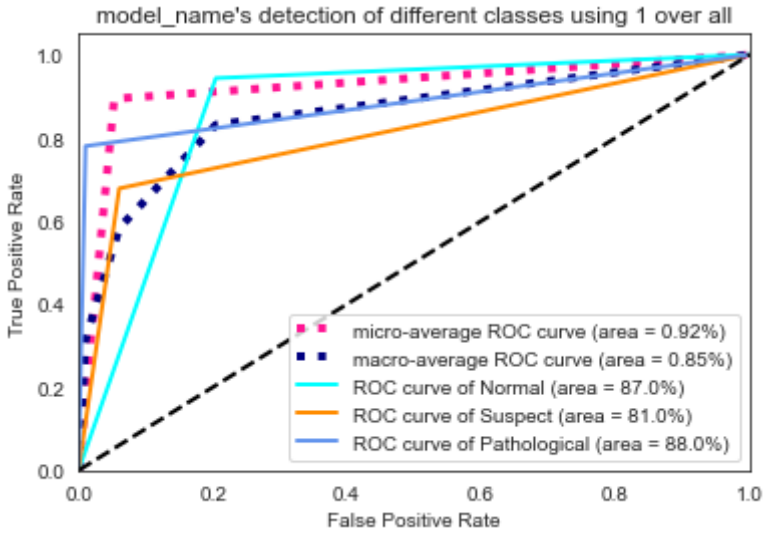





```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.94343434 0.05050505 0.00606061]
 [0.28735632 0.67816092 0.03448276]
 [0.06       0.16       0.78      ]]
```

```
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8666666666666667, 'recall': 0.78, 'f1-score': 0.8210526315789474, 'support': 50}
```






```
Best Parameters:
{}

************************************************************
```

2.2.2 OneVsRest Logistic Resgression Gridsearch

In [71]:

```
OVR_lg_gscv_params = {
    'OVR_lg__estimator__C': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'OVR_lg__estimator__penalty' : ['l1', 'l2', 'elasticnet'],
    'OVR_lg__estimator__solver' : ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga'],
    'OVR_lg__estimator__class_weight': [y_weights, 'balanced'],
    'OVR_lg__estimator__multi_class' : ['auto', 'multinomial']}
```

executed in 2ms, finished 22:44:17 2022-07-05

In [72]:

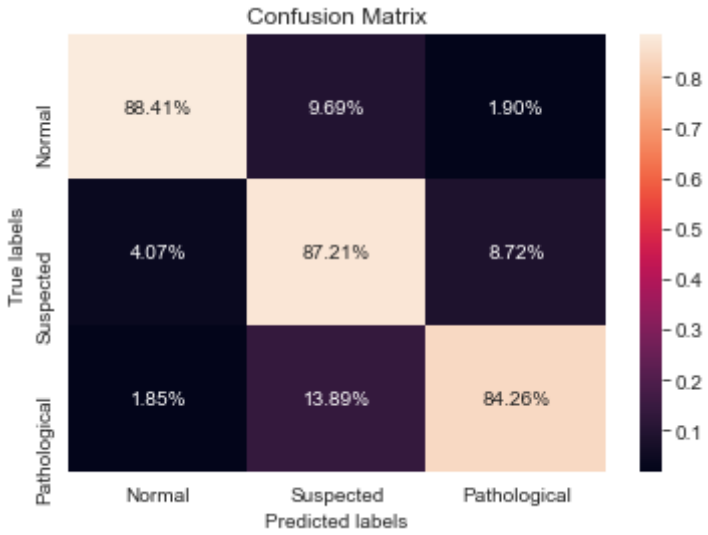
```
grid = GridSearchCV(OVR_lg_pipeline, OVR_lg_gscv_params, scoring = 'recall_micro')
```

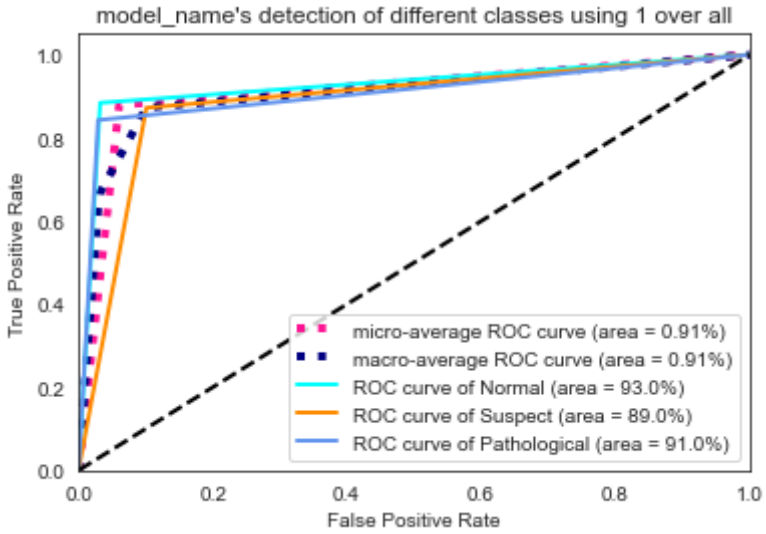
executed in 2ms, finished 22:44:17 2022-07-05

In [73]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'OVR_lg_gscv')
```

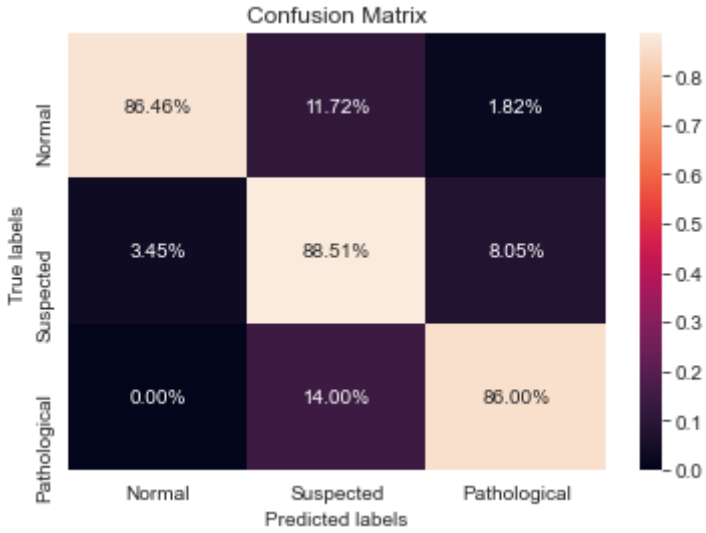
executed in 44.0s, finished 22:45:01 2022-07-05

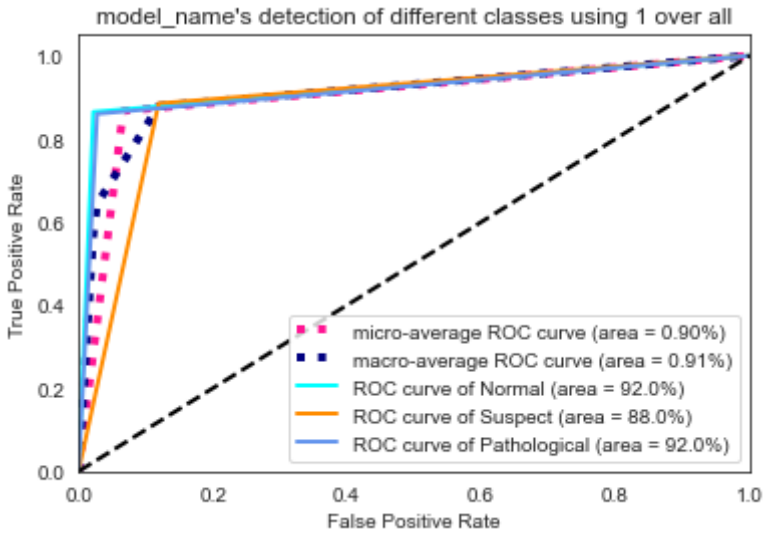
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.88411588 0.0969031  0.01898102]
 [0.04069767 0.87209302 0.0872093 ]
 [0.01851852 0.13888889 0.84259259]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.728, 'recall': 0.8425925925925926, 'f1-score': 0.7811158798283262, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.86464646 0.11717172 0.01818182]
 [0.03448276 0.88505747 0.08045977]
 [0.         0.14       0.86      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.7288135593220338, 'recall': 0.86, 'f1-score': 0.7889908256880733, 'support': 50}
```






```
Best Parameters:
{'OVR_lg__estimator__C': 2, 'OVR_lg__estimator__class_weight': 'balanced', 'OVR_lg__estimator__multi_class': 'auto', 'O
VR_lg__estimator__penalty': 'l1', 'OVR_lg__estimator__solver': 'liblinear'}

**********************************************************
```

Observations:

This model doesn't seem to be over fit. It performs better once it's done it's gridsearch for the best hyper-parameters.

2.3 K Nearest Neighbors

2.3.1 KNN Baseline

JUSTIFICATION: Should this model prove to be reliable, it would provide very quick results and not cost hospitals much computation.

In [74]:

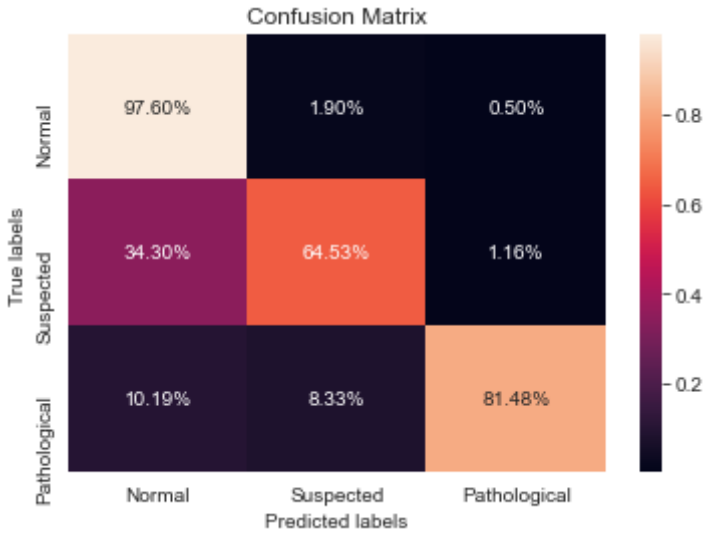
```
KNN_pipeline = Pipeline([
    ('KNN', KNeighborsClassifier())

])
baseline_params = {}

grid = GridSearchCV(KNN_pipeline, baseline_params, scoring = 'recall_micro')
```

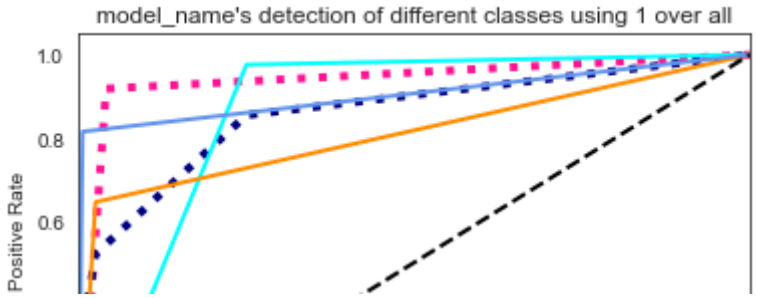
executed in 2ms, finished 22:45:01 2022-07-05

In [75]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'KNN_pipeline')
```

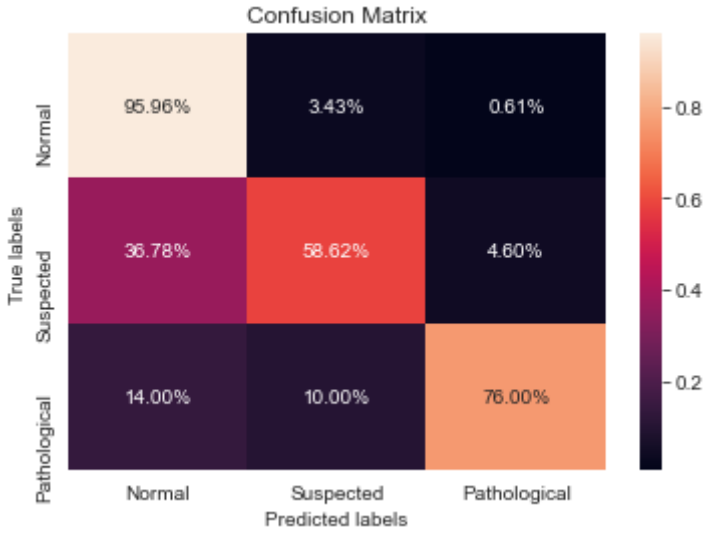
executed in 526ms, finished 22:45:01 2022-07-05

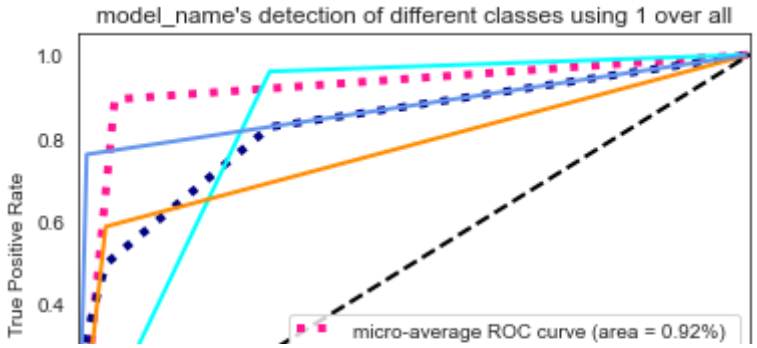
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.97602398 0.01898102 0.004995  ]
 [0.34302326 0.64534884 0.01162791]
 [0.10185185 0.08333333 0.81481481]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9263157894736842, 'recall': 0.8148148148148148, 'f1-score': 0.8669950738916257, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
******************************************************
[[0.95959596 0.03434343 0.00606061]
 [0.36781609 0.5862069  0.04597701]
 [0.14       0.1        0.76      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8444444444444444, 'recall': 0.76, 'f1-score': 0.8, 'support': 50}
```






```
Best Parameters:
{}

************************************************************
```

2.3.2 KNN Gridsearch

In [76]:

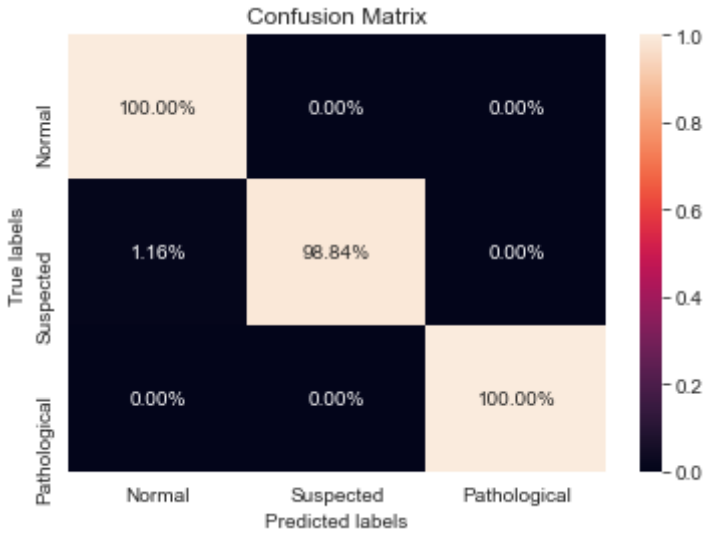
```
KNN_gscv_params = {
    'KNN__n_neighbors': list(range(1, 26, 2)),
    'KNN__metric' : ['euclidean', 'manhattan', 'chebyshev', 'minkowski'],
    'KNN__p' : [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    'KNN__weights' : ['uniform', 'distance']}

grid = GridSearchCV(KNN_pipeline, KNN_gscv_params, scoring = 'recall_micro')
```

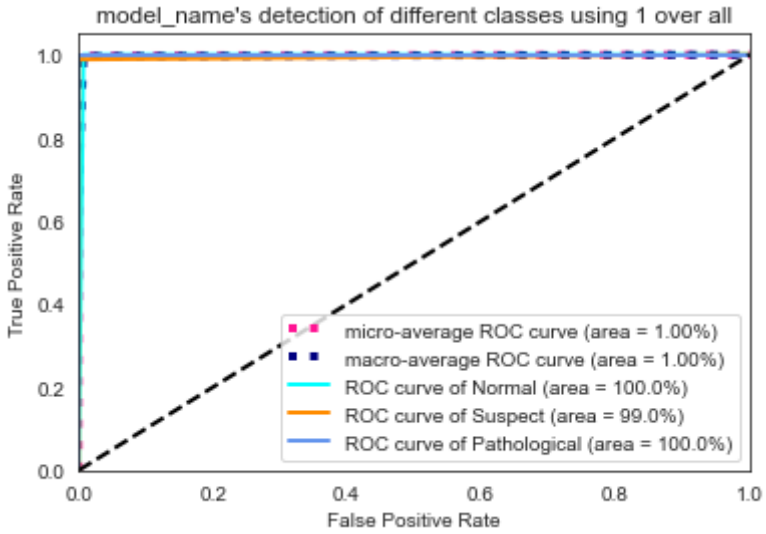
executed in 2ms, finished 22:45:02 2022-07-05

In [77]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'KNN_gscv')
```

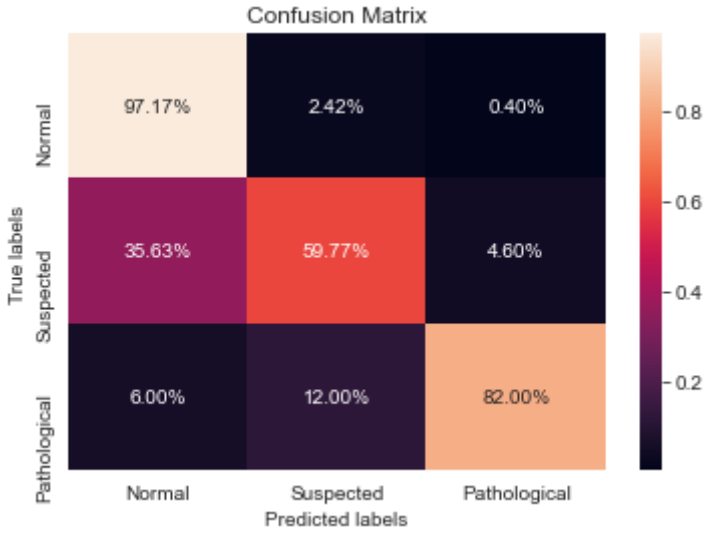
executed in 41.1s, finished 22:45:43 2022-07-05

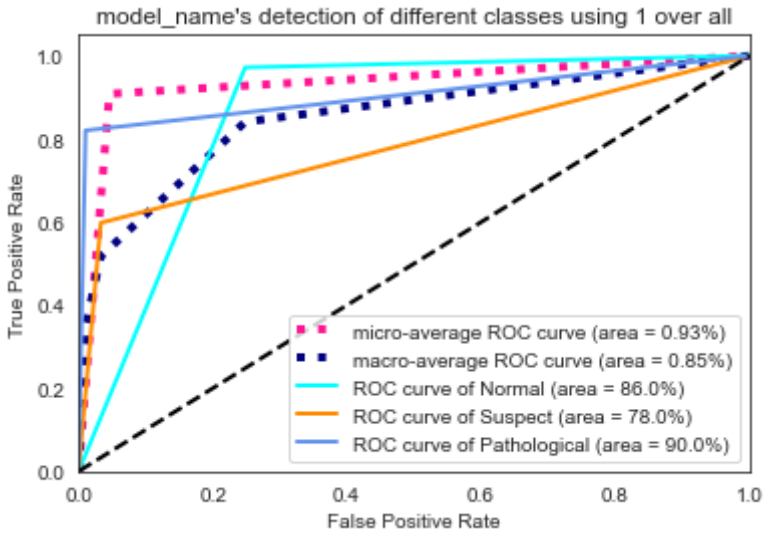
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[1.         0.         0.        ]
 [0.01162791 0.98837209 0.        ]
 [0.         0.         1.        ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.97171717 0.02424242 0.0040404 ]
 [0.35632184 0.59770115 0.04597701]
 [0.06       0.12       0.82      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8723404255319149, 'recall': 0.82, 'f1-score': 0.8453608247422681, 'support': 50}
```






```
Best Parameters:
{'KNN__metric': 'manhattan', 'KNN__n_neighbors': 3, 'KNN__p': 0.001, 'KNN__weights': 'distance'}

**********************************************************
```

Observations:

This model looks to be very overfit. In future work, I'd like to come back and better tune the hyper parameters to see if I couldn't get a model that generalizes much better.

2.4 OneVsRest KNN

2.4.1 OneVsRest KNN Baseline

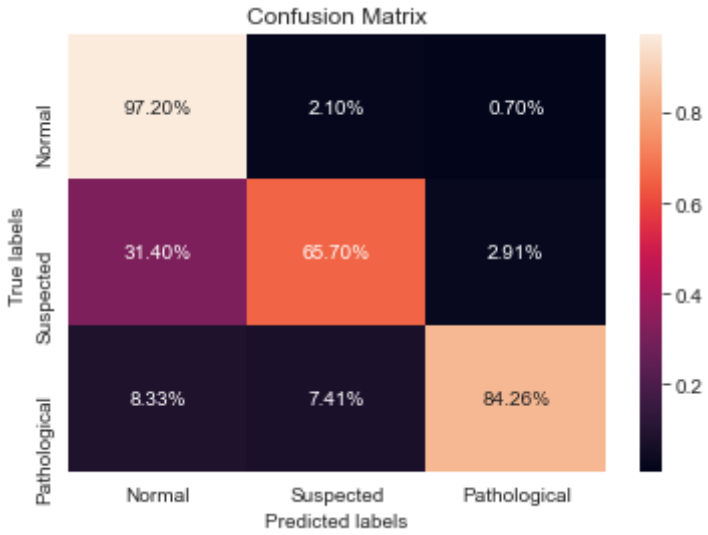
In [78]:
```
OVR_KNN_pipeline = Pipeline([
    ('OVR_KNN', OneVsRestClassifier(KNeighborsClassifier())),

])
baseline_params = {}

grid = GridSearchCV(OVR_KNN_pipeline, baseline_params, scoring = 'recall_micro')
```

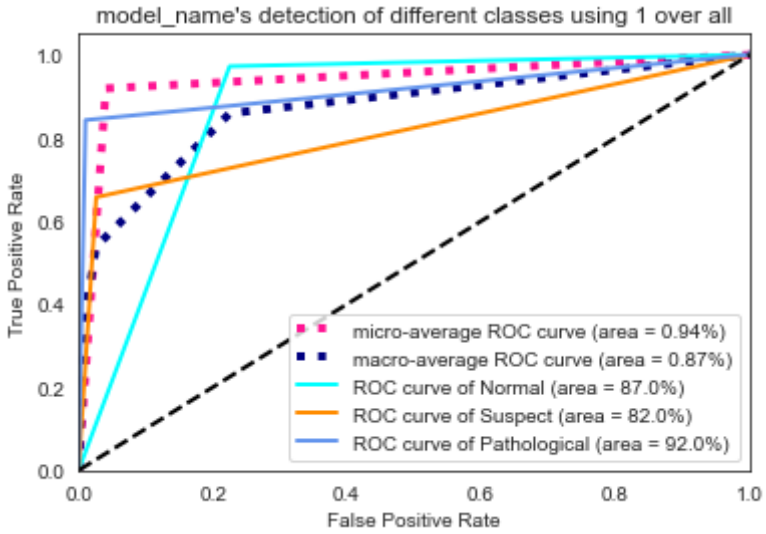
executed in 3ms, finished 22:45:43 2022-07-05

In [79]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'OVR_KNN_pipeline')
```

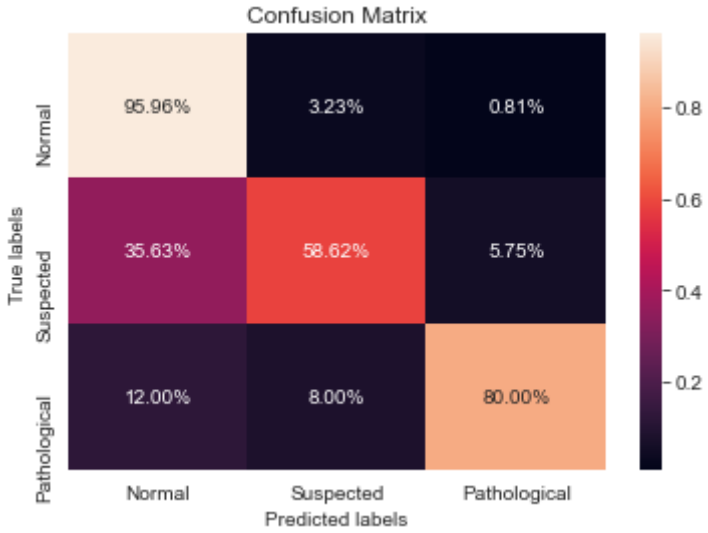
executed in 583ms, finished 22:45:43 2022-07-05

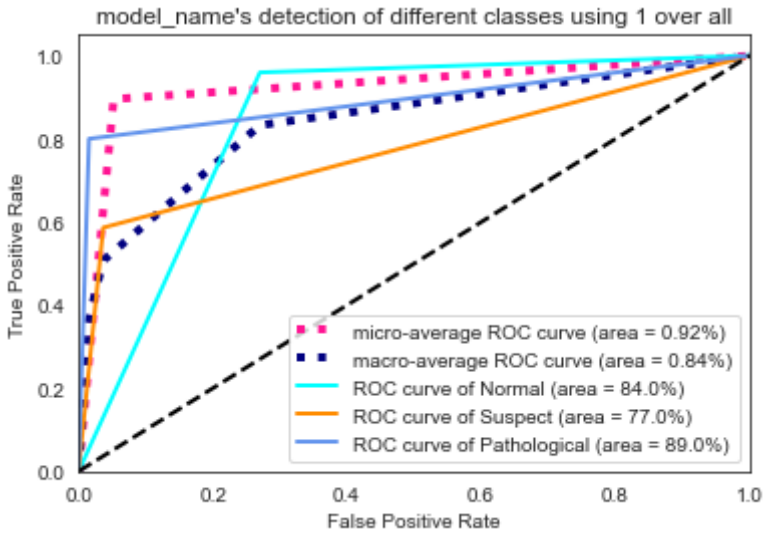
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.97202797 0.02097902 0.00699301]
 [0.31395349 0.65697674 0.02906977]
 [0.08333333 0.07407407 0.84259259]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.883495145631068, 'recall': 0.8425925925925926, 'f1-score': 0.8625592417061612, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.95959596 0.03232323 0.00808081]
 [0.35632184 0.5862069  0.05747126]
 [0.12       0.08       0.8       ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8163265306122449, 'recall': 0.8, 'f1-score': 0.8080808080808082, 'support': 50}
```






```
Best Parameters:
{}

************************************************************
```

2.4.2 OneVsRest KNN Gridsearch

In [80]:

```
OVR_KNN_params = {
    'OVR_KNN__estimator__n_neighbors': list(range(1, 26, 2)),
    'OVR_KNN__estimator__metric' : ['euclidean', 'manhattan', 'chebyshev', 'minkowski'],
    'OVR_KNN__estimator__weights' : ['uniform', 'distance']}

grid = GridSearchCV(OVR_KNN_pipeline, OVR_KNN_params, scoring = 'recall_micro')
```

executed in 2ms, finished 22:45:43 2022-07-05

In [81]:

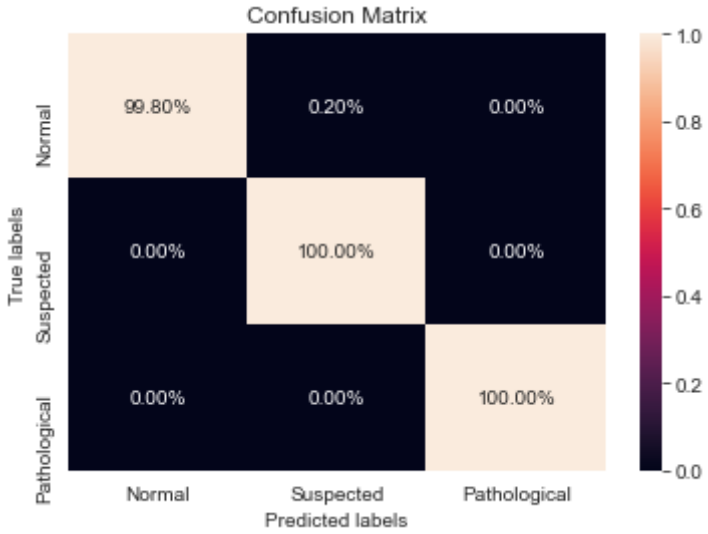
```
grid = GridSearchCV(OVR_KNN_pipeline, OVR_KNN_params, scoring = 'recall_micro')
```

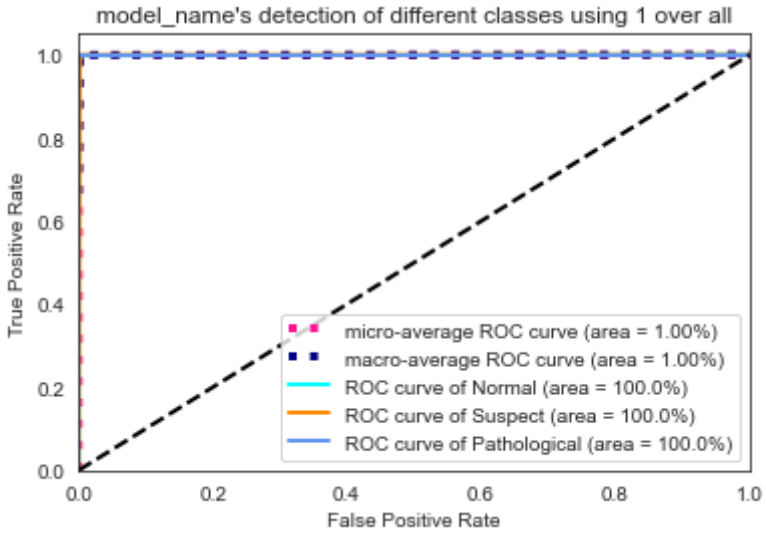
executed in 2ms, finished 22:45:43 2022-07-05

In [82]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'OVR_KNN_gscv')
```

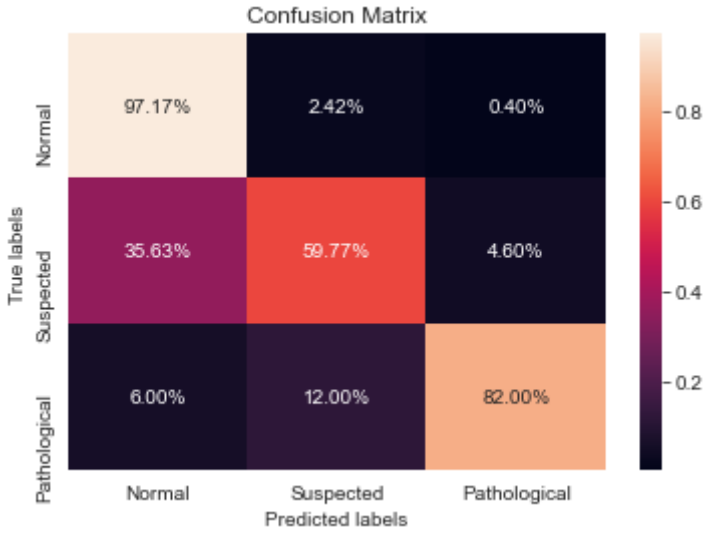
executed in 9.55s, finished 22:45:53 2022-07-05

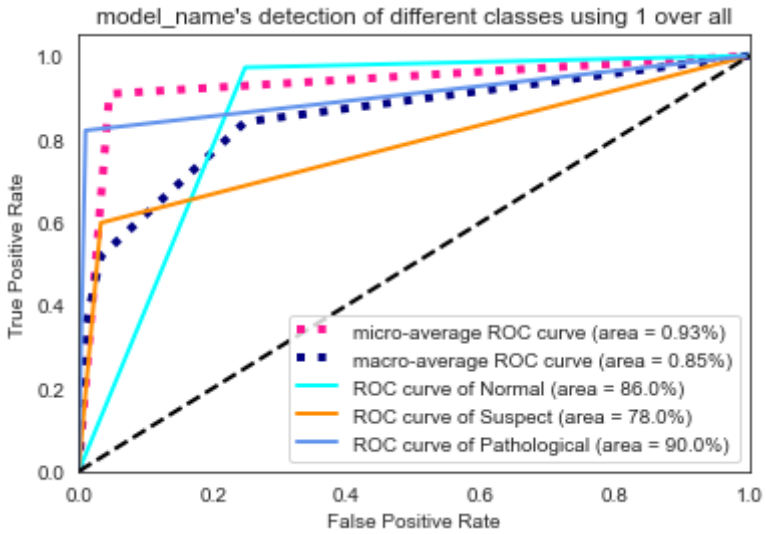
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.998002 0.001998 0.      ]
 [0.       1.       0.      ]
 [0.       0.       1.      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.97171717 0.02424242 0.0040404 ]
 [0.35632184 0.59770115 0.04597701]
 [0.06       0.12       0.82      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8723404255319149, 'recall': 0.82, 'f1-score': 0.8453608247422681, 'support': 50}
```






```
Best Parameters:
{'OVR_KNN__estimator__metric': 'manhattan', 'OVR_KNN__estimator__n_neighbors': 3, 'OVR_KNN__estimator__weights': 'dista
nce'}

**********************************************************
```

Observations:

This model looks to be very overfit. In future work, I'd like to come back and better tune the hyper parameters to see if I couldn't get a model that generalizes much better.

2.5 Support Vector Machine

JUSTIFICATION: SVM are a natural choice for a multiclass problem. They're limited by their runtime however.

2.5.1 Support Vector Machine Baseline

In [83]:

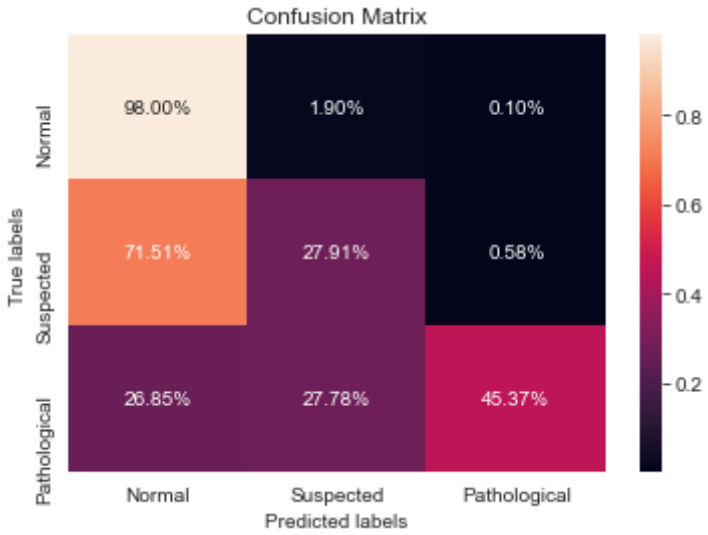
```
SVC_pipeline = Pipeline([
    ('SVC', svm.SVC())

])
baseline_params = {'SVC__random_state' : [42]}

grid = GridSearchCV(SVC_pipeline, baseline_params, scoring = 'recall_micro')
```

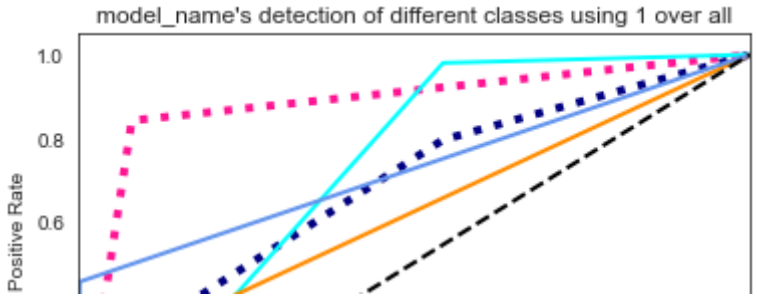
executed in 2ms, finished 22:45:53 2022-07-05

In [84]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'SVC_pipeline')
```

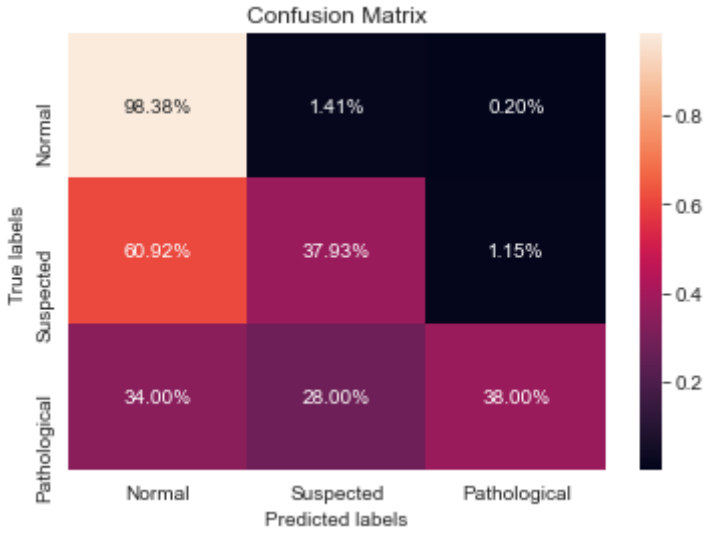
executed in 534ms, finished 22:45:53 2022-07-05

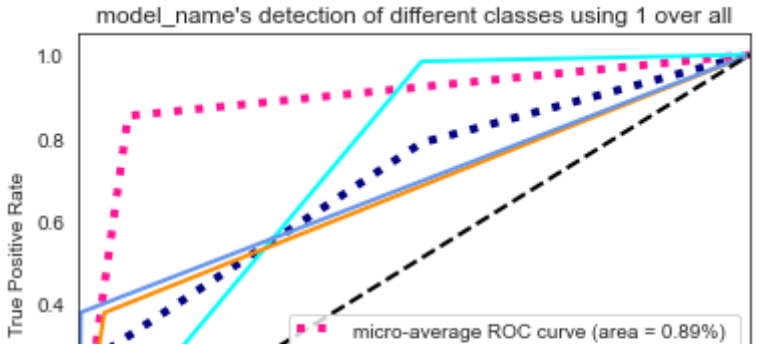
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.98001998 0.01898102 0.000999  ]
 [0.71511628 0.27906977 0.00581395]
 [0.26851852 0.27777778 0.4537037 ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9607843137254902, 'recall': 0.4537037037037037, 'f1-score': 0.6163522012578616, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
******************************************************
[[0.98383838 0.01414141 0.0020202 ]
 [0.6091954  0.37931034 0.01149425]
 [0.34       0.28       0.38      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9047619047619048, 'recall': 0.38, 'f1-score': 0.5352112676056339, 'support': 50}
```






```
Best Parameters:
{'SVC__random_state': 42}

************************************************************
```

Observations:

This model looks to be very overfit. In future work, I'd like to come back and better tune the hyper parameters to see if I couldn't get a model that generalizes much better.

2.5.2 Support Vector Machine Gridsearch

In [85]:

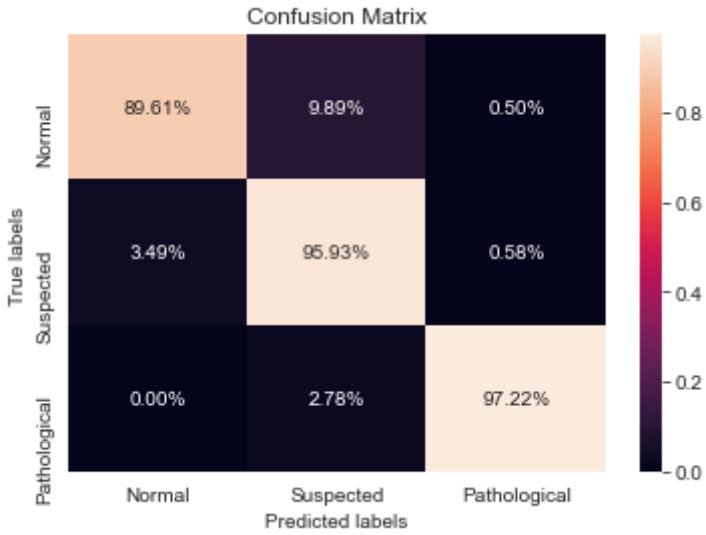
```
SVC_gscv_params = {
    'SVC__C': [ 3450, 3500, 3550, 3600],
    'SVC__kernel' : ['poly',],
                    #'linear', 'rbf',],
    'SVC__class_weight': [y_weights, 'balanced']}

grid = GridSearchCV(SVC_pipeline, SVC_gscv_params, scoring = 'recall_micro')
```

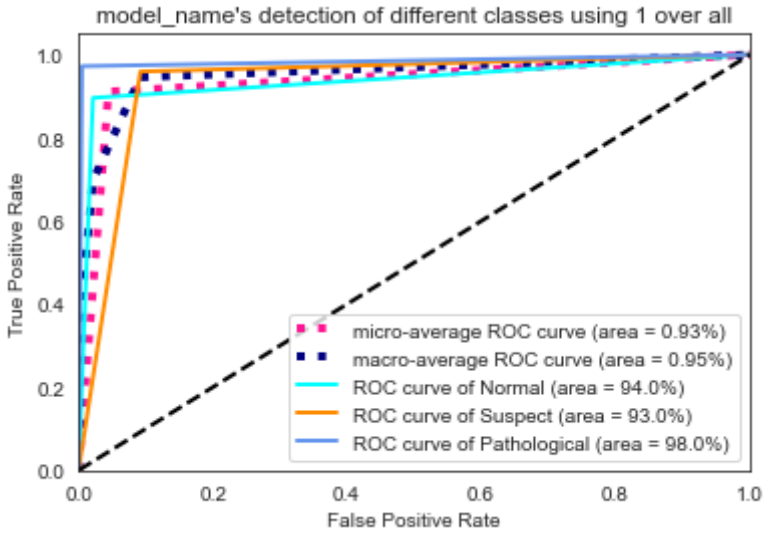
executed in 2ms, finished 22:45:53 2022-07-05

In [86]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'SVC_gscv')
```

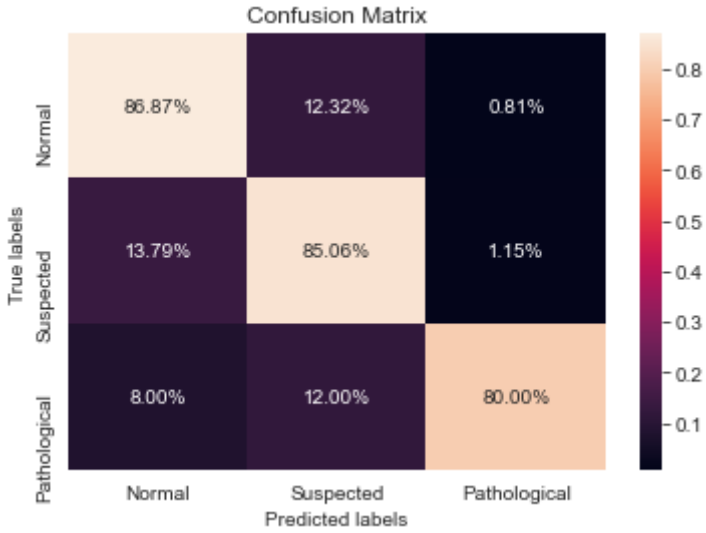
executed in 11.3s, finished 22:46:05 2022-07-05

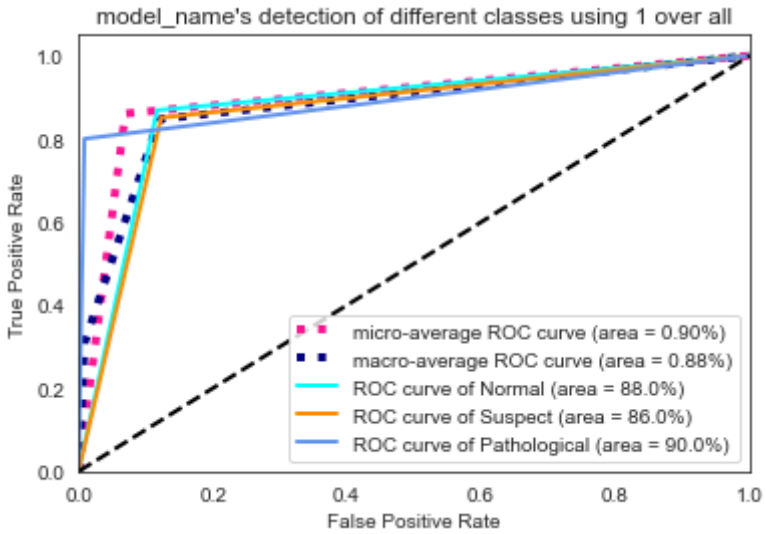
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.8961039  0.0989011  0.004995  ]
 [0.03488372 0.95930233 0.00581395]
 [0.         0.02777778 0.97222222]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9459459459459459, 'recall': 0.9722222222222222, 'f1-score': 0.9589041095890412, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.86868687 0.12323232 0.00808081]
 [0.13793103 0.85057471 0.01149425]
 [0.08       0.12       0.8       ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8888888888888888, 'recall': 0.8, 'f1-score': 0.8421052631578948, 'support': 50}
```






```
Best Parameters:
{'SVC__C': 3600, 'SVC__class_weight': 'balanced', 'SVC__kernel': 'poly'}

************************************************************
```

Observations:

This model looks to be very overfit. In future work, I'd like to come back and better tune the hyper parameters to see if I couldn't get a model that generalizes much better. Gridsearch improved the performance of the Support Vector Machine a lot, which is expected since SVM are very sensitive to class imbalance.

2.6 OneVsRest Support Vector Machine

2.6.1 OneVsRest Support Vector Machine Baseline

In [87]:

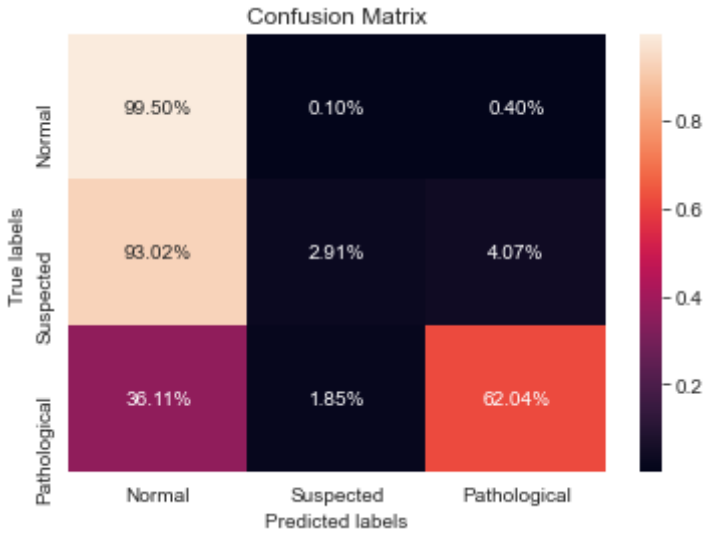
```
OVR_SVC_pipeline = Pipeline([
    ('OVR_SVC', OneVsRestClassifier(svm.SVC(random_state = 42))),

])
baseline_params = {}

grid = GridSearchCV(OVR_SVC_pipeline, baseline_params, scoring = 'recall_micro')
```

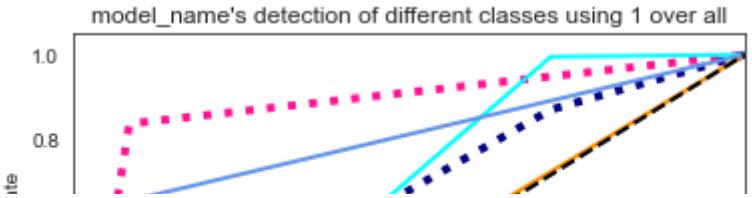
executed in 2ms, finished 22:46:05 2022-07-05

In [88]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'OVR_SVC_pipeline')
```

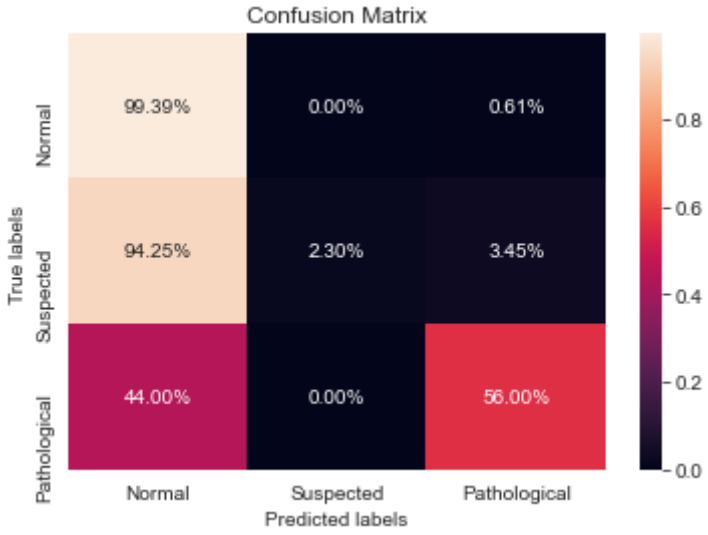
executed in 660ms, finished 22:46:05 2022-07-05

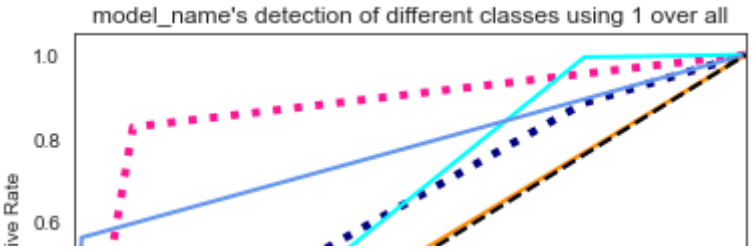
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.995005   0.000999   0.003996  ]
 [0.93023256 0.02906977 0.04069767]
 [0.36111111 0.01851852 0.62037037]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8589743589743589, 'recall': 0.6203703703703703, 'f1-score': 0.7204301075268816, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.99393939 0.         0.00606061]
 [0.94252874 0.02298851 0.03448276]
 [0.44       0.         0.56      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8235294117647058, 'recall': 0.56, 'f1-score': 0.6666666666666666, 'support': 50}
```






```
Best Parameters:
{}

************************************************************
```

2.6.2 OneVsRest Support Vector Machine Gridsearch

In [89]:

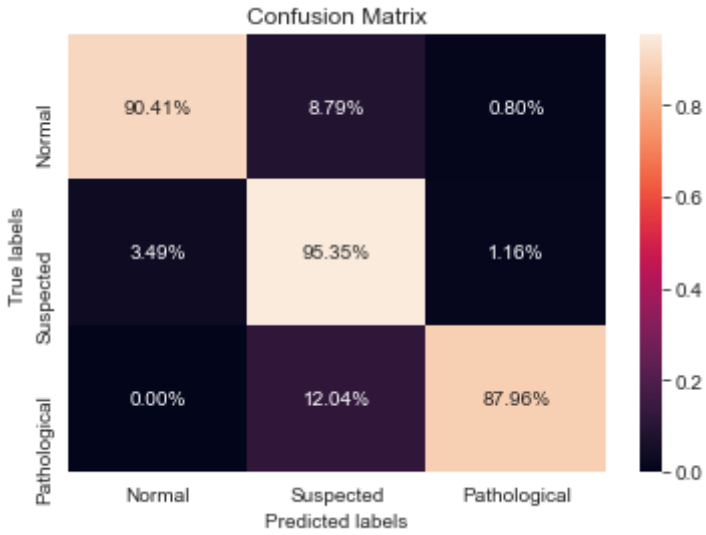
```
OVR_SVM_gscv_params = {
    'OVR_SVC__estimator__C': [3000, 3250, 3500, 3750, 4000,],
    'OVR_SVC__estimator__kernel': ['poly', 'linear', 'rbf',],
    'OVR_SVC__estimator__class_weight': [y_weights, 'balanced']}

grid = GridSearchCV(OVR_SVC_pipeline, param_grid = OVR_SVM_gscv_params,
                            scoring = 'recall_micro')
```

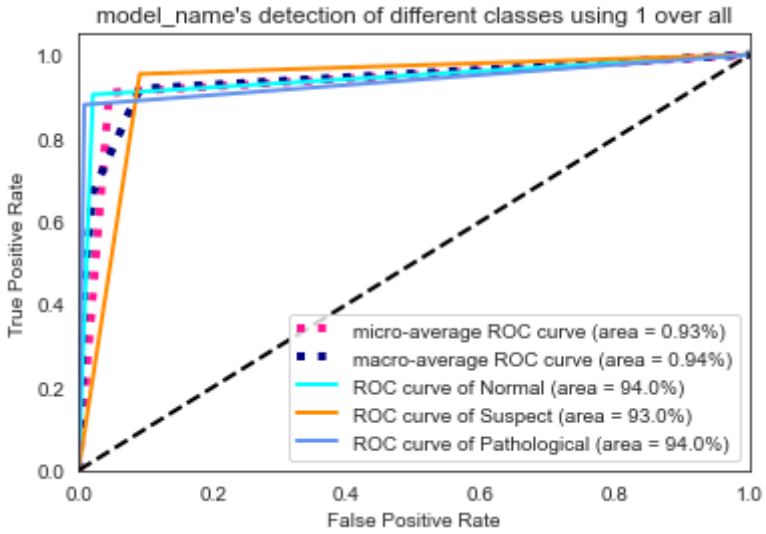
executed in 2ms, finished 22:46:05 2022-07-05

In [90]:
```
show_train_and_test(grid, X_train, y_train, X_val, 'OVR_SVM_gscv')
```

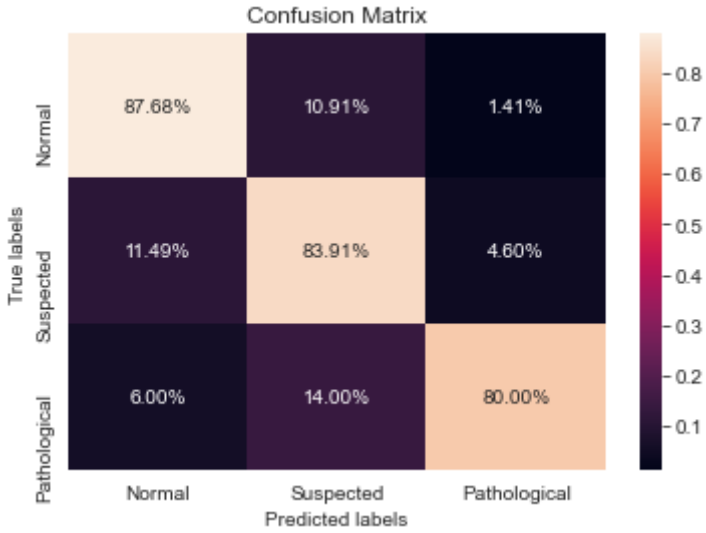
executed in 18m 54s, finished 23:04:59 2022-07-05

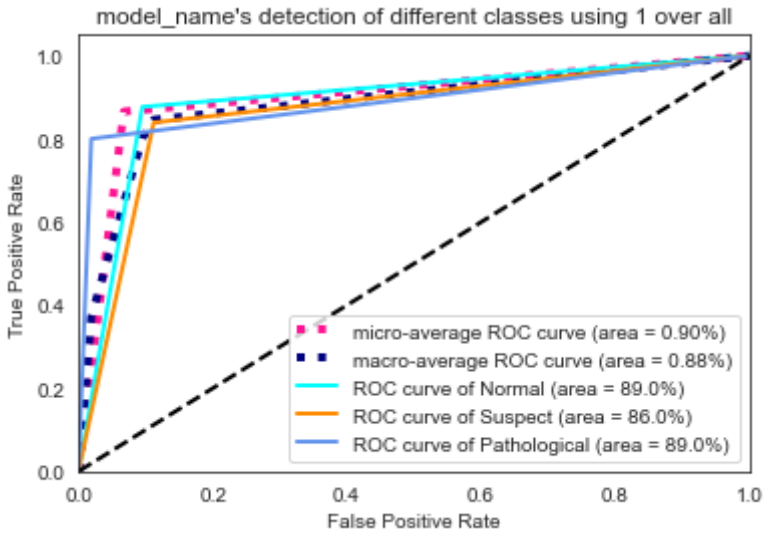
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.9040959  0.08791209 0.00799201]
 [0.03488372 0.95348837 0.01162791]
 [0.         0.12037037 0.87962963]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9047619047619048, 'recall': 0.8796296296296297, 'f1-score': 0.892018779342723, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.87676768 0.10909091 0.01414141]
 [0.11494253 0.83908046 0.04597701]
 [0.06       0.14       0.8       ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.7843137254901961, 'recall': 0.8, 'f1-score': 0.792079207920792, 'support': 50}
```






```
Best Parameters:
{'OVR_SVC__estimator__C': 3250, 'OVR_SVC__estimator__class_weight': 'balanced', 'OVR_SVC__estimator__kernel': 'poly'}

**********************************************************
```

Observations:

This model looks to be very overfit. In future work, I'd like to come back and better tune the hyper parameters to see if I couldn't get a model that generalizes much better.

2.7 Random Forest:

JUSTIFICATION: Random Forest is another natural choice for a multiclass problem and this will run much faster than a SVM.

2.7.1 Random Forest Baseline

In [91]:

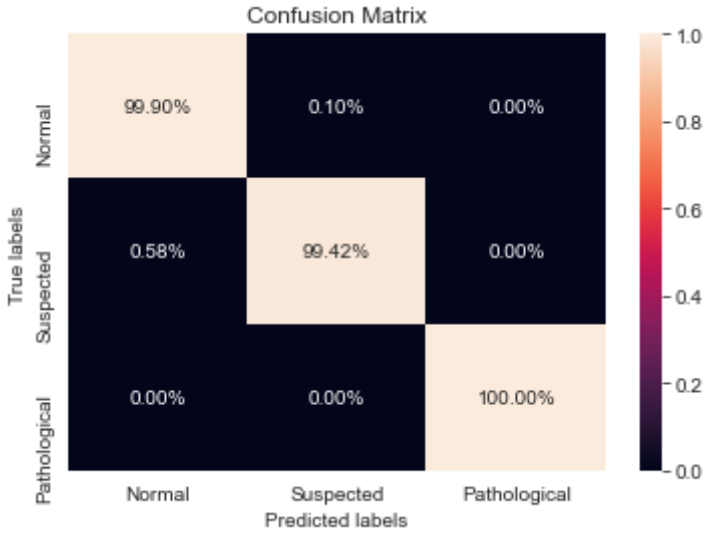
```
rf_pipeline = Pipeline([
    ('rf', RandomForestClassifier())

])
baseline_params = {'rf__random_state' : [42]}

grid = GridSearchCV(rf_pipeline, baseline_params, scoring = 'recall_micro')
```

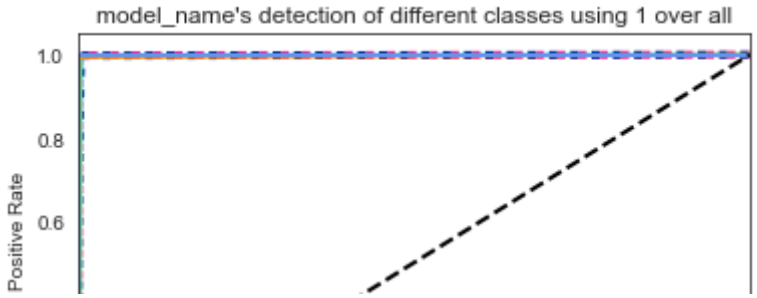
executed in 2ms, finished 23:04:59 2022-07-05

In [92]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'rf_pipeline')
```

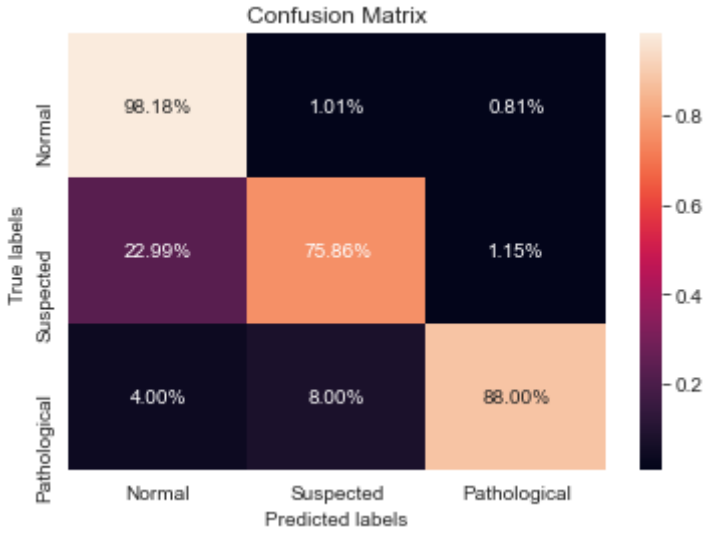
executed in 1.31s, finished 23:05:00 2022-07-05

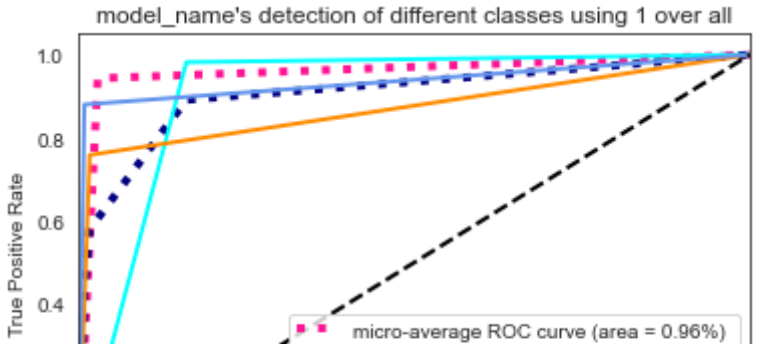
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[9.99000999e-01 9.99000999e-04 0.00000000e+00]
 [5.81395349e-03 9.94186047e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
******************************************************
[[0.98181818 0.01010101 0.00808081]
 [0.22988506 0.75862069 0.01149425]
 [0.04       0.08       0.88      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.8979591836734694, 'recall': 0.88, 'f1-score': 0.888888888888889, 'support': 50}
```






```
Best Parameters:
{'rf__random_state': 42}

************************************************************
```

Observations:

random forest fits very well, but it has a tendency to overfit drastically, as we can see from a baseline model.

2.7.2 Random Forest Gridsearch

In [93]:

```
rf_gscv_params = {
    'rf__min_samples_split': [4, 5, 6, 7, 8, 9, 10],
    'rf__min_samples_leaf' : [1, 2, 3, 4, 5, 6],
    'rf__max_features' : ['auto', 'sqrt', 'log2'],
    'rf__criterion': ["gini", "entropy"],
    'rf__class_weight': ['balanced', y_weights]}

grid = GridSearchCV(rf_pipeline, rf_gscv_params, scoring = 'recall_micro')
```

executed in 2ms, finished 23:05:00 2022-07-05

In [94]:

```
show_train_and_test(grid, X_train, y_train, X_val, 'rf_gscv')
```

executed in 2m 49s, finished 23:07:49 2022-07-05

```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.98801199 0.00999001 0.001998  ]
 [0.         1.         0.        ]
 [0.         0.         1.        ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9818181818181818, 'recall': 1.0, 'f1-score': 0.9908256880733944, 'support': 108}
```

Observations:

This model looks to be very overfit. In future work, I'd like to come back and better tune the hyper parameters to see if I couldn't get a model that generalizes much better.

2.8 OneVsRest Random Forest

2.8.1 OneVsRest Random Forest Baseline

In [170]:

```
OVR_rf_pipeline = Pipeline([
    ('OVR_rf', OneVsRestClassifier(RandomForestClassifier(random_state = 42))),

])
baseline_params = {}

grid = GridSearchCV(OVR_rf_pipeline, baseline_params, scoring = 'recall_micro')
```

executed in 8ms, finished 10:04:06 2022-07-06

In [171]:
```
grid.get_params()
```

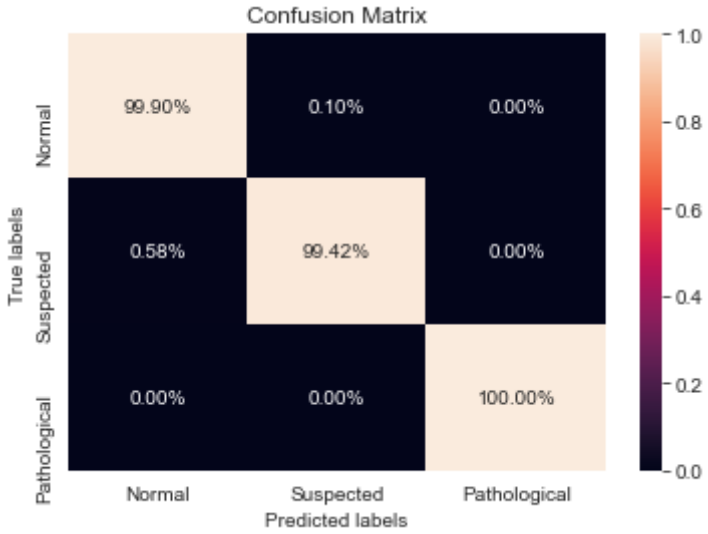
executed in 26ms, finished 10:04:14 2022-07-06

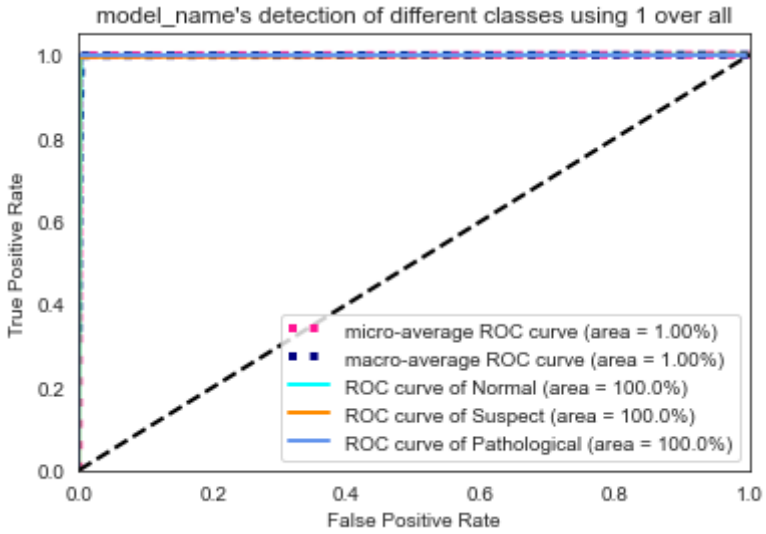
```
{'cv': None,
 'error_score': nan,
 'estimator__memory': None,
 'estimator__steps': [('OVR_rf',
   OneVsRestClassifier(estimator=RandomForestClassifier(random_state=42)))],
 'estimator__verbose': False,
 'estimator__OVR_rf': OneVsRestClassifier(estimator=RandomForestClassifier(random_state=42)),
 'estimator__OVR_rf__estimator__bootstrap': True,
 'estimator__OVR_rf__estimator__ccp_alpha': 0.0,
 'estimator__OVR_rf__estimator__class_weight': None,
 'estimator__OVR_rf__estimator__criterion': 'gini',
 'estimator__OVR_rf__estimator__max_depth': None,
 'estimator__OVR_rf__estimator__max_features': 'auto',
 'estimator__OVR_rf__estimator__max_leaf_nodes': None,
 'estimator__OVR_rf__estimator__max_samples': None,
 'estimator__OVR_rf__estimator__min_impurity_decrease': 0.0,
 'estimator__OVR_rf__estimator__min_impurity_split': None,
 'estimator__OVR_rf__estimator__min_samples_leaf': 1,
 'estimator__OVR_rf__estimator__min_samples_split': 2,
 'estimator__OVR_rf__estimator__min_weight_fraction_leaf': 0.0,
 'estimator__OVR_rf__estimator__n_estimators': 100,
 'estimator__OVR_rf__estimator__n_jobs': None,
 'estimator__OVR_rf__estimator__oob_score': False,
 'estimator__OVR_rf__estimator__random_state': 42,
 'estimator__OVR_rf__estimator__verbose': 0,
 'estimator__OVR_rf__estimator__warm_start': False,
 'estimator__OVR_rf__estimator': RandomForestClassifier(random_state=42),
 'estimator__OVR_rf__n_jobs': None,
 'estimator': Pipeline(steps=[('OVR_rf',
                 OneVsRestClassifier(estimator=RandomForestClassifier(random_state=42)))]),
 'iid': 'deprecated',
 'n_jobs': None,
 'param_grid': {},
 'pre_dispatch': '2*n_jobs',
 'refit': True,
 'return_train_score': False,
 'scoring': 'recall_micro',
 'verbose': 0}
```

In [96]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'OVR_rf_pipeline')
```

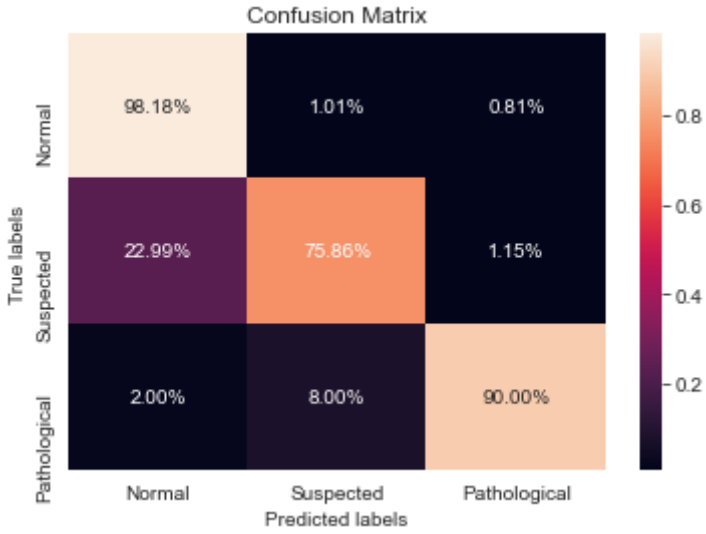
executed in 2.81s, finished 23:07:52 2022-07-05

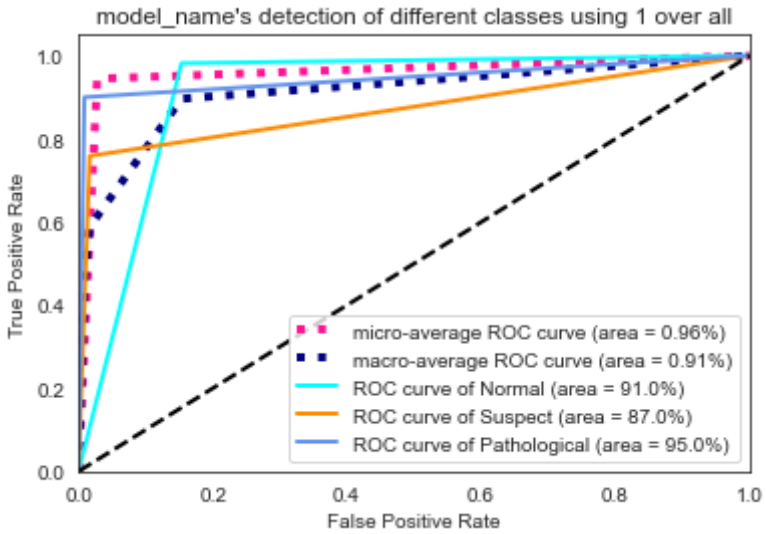
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[9.99000999e-01 9.99000999e-04 0.00000000e+00]
 [5.81395349e-03 9.94186047e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 1.0, 'recall': 1.0, 'f1-score': 1.0, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.98181818 0.01010101 0.00808081]
 [0.22988506 0.75862069 0.01149425]
 [0.02       0.08       0.9       ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9, 'recall': 0.9, 'f1-score': 0.9, 'support': 50}
```






```
Best Parameters:
{}

************************************************************
```

2.8.2 OneVsRest Random Forrest Gridsearch

In [174]:

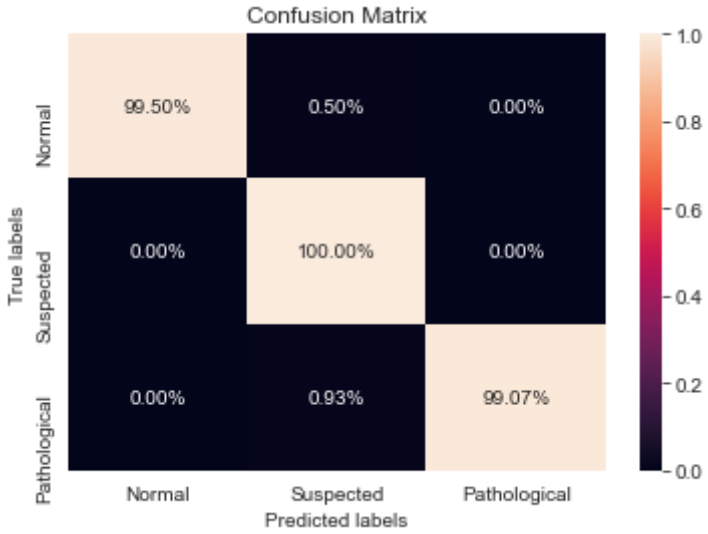
```
OVR_rf_gscv_params = {
    'OVR_rf__estimator__min_samples_split': [4, 5, 6, 7, 8, 9, 10],
    'OVR_rf__estimator__min_samples_leaf' : [1, 2, 3, 4, 5, 6],
    'OVR_rf__estimator__max_features' : ['auto', 'sqrt', 'log2'],
    'OVR_rf__estimator__criterion': ["gini", "entropy"],
    'OVR_rf__estimator__class_weight': ['balanced', y_weights]}

grid = GridSearchCV(OVR_rf_pipeline, OVR_rf_gscv_params,
                            scoring = 'recall_micro')
```

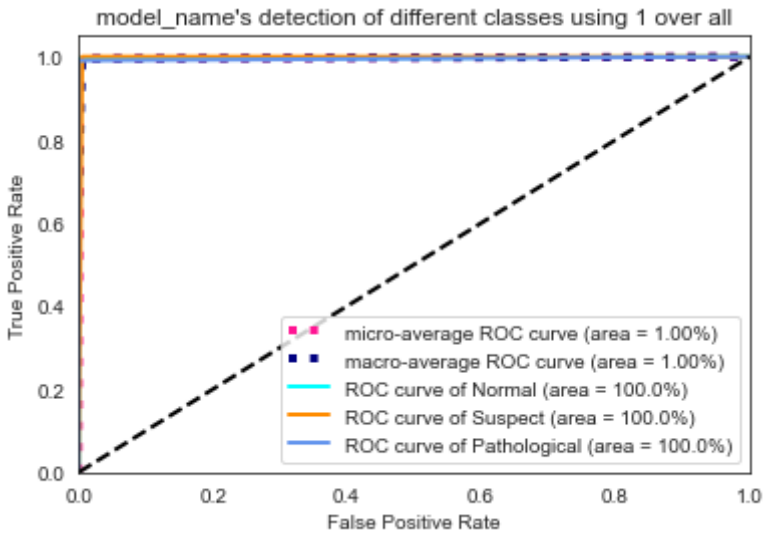
executed in 8ms, finished 10:05:42 2022-07-06

In [98]:
```
1 show_train_and_test(grid, X_train, y_train, X_val, 'OVR_rf_gscv')
```

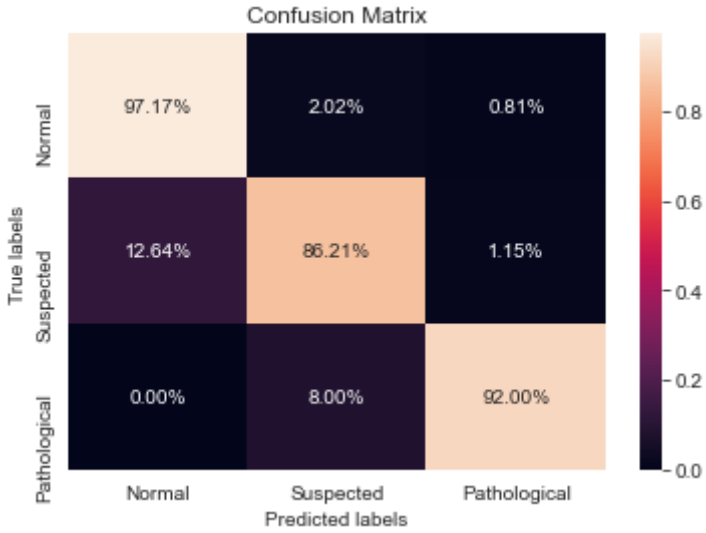
executed in 7m 39s, finished 23:15:31 2022-07-05

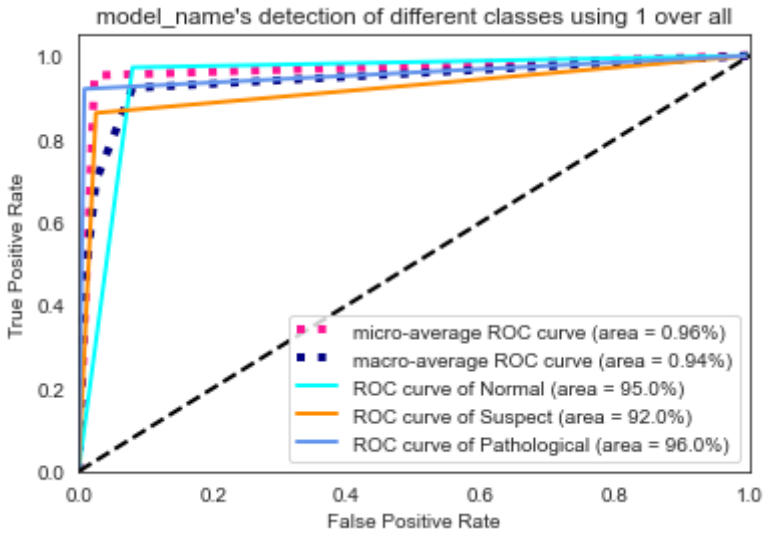
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.995005   0.004995   0.        ]
 [0.         1.         0.        ]
 [0.         0.00925926 0.99074074]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 1.0, 'recall': 0.9907407407407407, 'f1-score': 0.9953488372093023, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.97171717 0.02020202 0.00808081]
 [0.12643678 0.86206897 0.01149425]
 [0.         0.08       0.92      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9019607843137255, 'recall': 0.92, 'f1-score': 0.9108910891089109, 'support': 50}
```






```
Best Parameters:
{'OVR_rf__estimator__class_weight': 'balanced', 'OVR_rf__estimator__criterion': 'entropy', 'OVR_rf__estimator__max_feat
ures': 'auto', 'OVR_rf__estimator__min_samples_leaf': 1, 'OVR_rf__estimator__min_samples_split': 8}

**********************************************************
```

Observations: This model looks to be very overfit. In future work, I'd like to come back and better tune the hyper parameters to see if I couldn't get a model that generalizes much better.

3 MODEL ANALYSIS

3.1 Analyzing Model Performances

In [99]:

```
machine_labels = ["Logistic Regression Baseline",
                  "Logistic Regression Gridsearch",
                  "OneVsRest Logistic Regression Baseline",
                 "OneVsRest Logistic Regression Gridsearch",
                  "KNN Baseline",
                 "KNN Gridsearch",
                  "OneVsRest KNN Gridsearch",
                 "OneVsRest KNN Gridsearch",
                  "SVM Baseline",
                 "SVM Gridsearch",
                  "OneVsRest SVM Baseline",
                 "OneVsRest SVM Gridsearch",
                  "Random Forest Baseline",
                 "Random Forest Gridsearch",
                  "OneVsRest Random Forest Baseline",
                 "OneVsRest Random Forest Gridsearch"]
```

executed in 2ms, finished 23:15:31 2022-07-05

In [100]:

```
reports_df = pd.DataFrame(reports)
reports_df_test = reports_df.loc[reports_df["support"] == 50]
reports_df_test["name"] = machine_labels
reports_df_test.set_index("name", inplace = True)
reports_df_test.sort_values(by = 'recall', ascending = False, inplace = True)
reports_df_test
```

executed in 16ms, finished 23:15:31 2022-07-05

name	precision	recall	f1-score	support
OneVsRest Random Forest Gridsearch	0.901961	0.92	0.910891	50
OneVsRest Random Forest Baseline	0.900000	0.90	0.900000	50
Random Forest Baseline	0.897959	0.88	0.888889	50
Random Forest Gridsearch	0.956522	0.88	0.916667	50
OneVsRest Logistic Regression Gridsearch	0.728814	0.86	0.788991	50
Logistic Regression Baseline	0.807692	0.84	0.823529	50
Logistic Regression Gridsearch	0.750000	0.84	0.792453	50
KNN Gridsearch	0.872340	0.82	0.845361	50
OneVsRest KNN Gridsearch	0.872340	0.82	0.845361	50
OneVsRest KNN Gridsearch	0.816327	0.80	0.808081	50
SVM Gridsearch	0.888889	0.80	0.842105	50
OneVsRest SVM Gridsearch	0.784314	0.80	0.792079	50
OneVsRest Logistic Regression Baseline	0.866667	0.78	0.821053	50
KNN Baseline	0.844444	0.76	0.800000	50
OneVsRest SVM Baseline	0.823529	0.56	0.666667	50
SVM Baseline	0.904762	0.38	0.535211	50

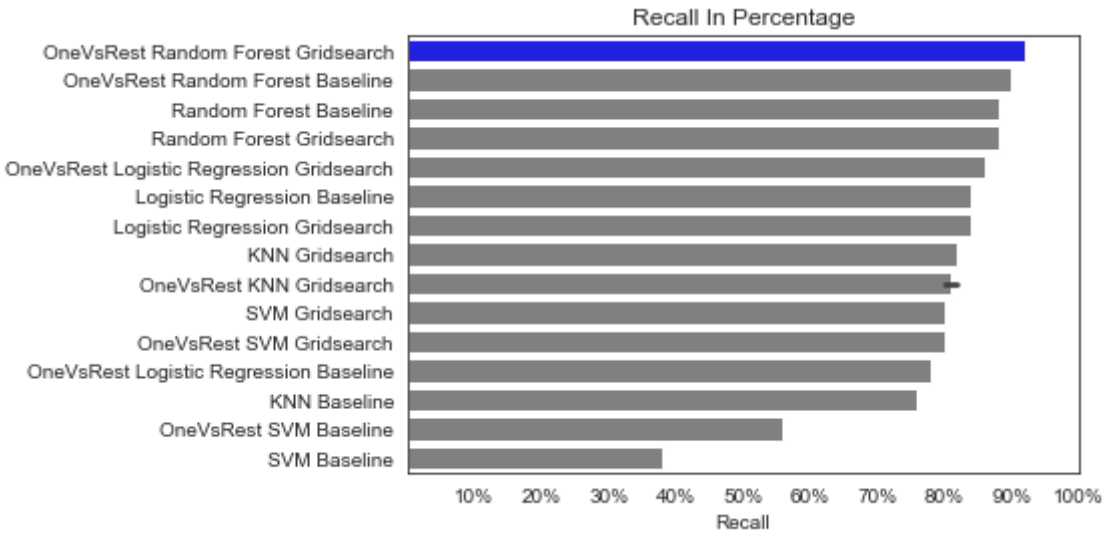
	CITATION: https://stackoverflow.com/questions/31074758/how-to-set-a-different-color-to-the-largest-bar-in-a-seaborn-barplot (https://stackoverflow.com/questions/31074758/how-to-set-a-different-color-to-the-largest-bar-in-a-seaborn-barplot)

In [236]:

```
clrs = ['grey' if (x < max (reports_df_test["recall"])) else 'blue' for x in reports_df_test["reca

model_performance_plot = sns.barplot(data = reports_df_test,
                                     y = reports_df_test.index,
                                    x = 'recall',
                                     orient = 'h',
                                    palette = clrs)

model_performance_plot.set_title("Recall In Percentage")
x_ticks = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
labels = ["10%", "20%", "30%", "40%", "50%", "60%", "70%", "80%", "90%", "100%"]
plt.xticks(ticks = x_ticks, labels = labels)
model_performance_plot.set_ylabel("")
model_performance_plot.set_xlabel("Recall")

#plt.tight_layout()
#plt.savefig("Machines Ranked by Recall in Pathological Class");
```

executed in 175ms, finished 11:43:33 2022-07-06

```
Text(0.5, 0, 'Recall')
```


Recall In Percentage
OneVsRest Random Forest Gridsearch
OneVsRest Random Forest Baseline
Random Forest Baseline
Random Forest Gridsearch
OneVsRest Logistic Regression Gridsearch
Logistic Regression Baseline
Logistic Regression Gridsearch
KNN Gridsearch
OneVsRest KNN Gridsearch
SVM Gridsearch
OneVsRest SVM Gridsearch
OneVsRest Logistic Regression Baseline
KNN Baseline
OneVsRest SVM Baseline
SVM Baseline
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
Recall

In [113]:
```
1 reports_df_test
```

executed in 25ms, finished 09:44:51 2022-07-06

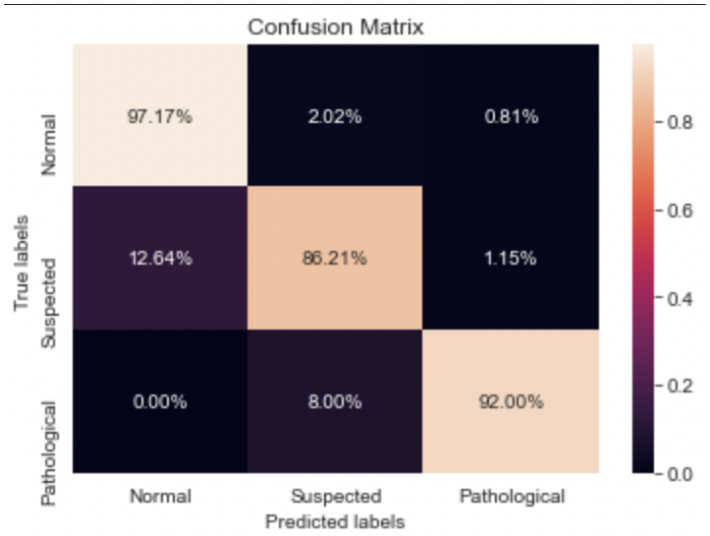
name	precision	recall	f1-score	support
OneVsRest Random Forest Gridsearch	0.901961	0.92	0.910891	50
OneVsRest Random Forest Baseline	0.900000	0.90	0.900000	50
Random Forest Baseline	0.897959	0.88	0.888889	50
Random Forest Gridsearch	0.956522	0.88	0.916667	50
OneVsRest Logistic Regression Gridsearch	0.728814	0.86	0.788991	50
Logistic Regression Baseline	0.807692	0.84	0.823529	50
Logistic Regression Gridsearch	0.750000	0.84	0.792453	50
KNN Gridsearch	0.872340	0.82	0.845361	50
OneVsRest KNN Gridsearch	0.872340	0.82	0.845361	50
OneVsRest KNN Gridsearch	0.816327	0.80	0.808081	50
SVM Gridsearch	0.888889	0.80	0.842105	50
OneVsRest SVM Gridsearch	0.784314	0.80	0.792079	50
OneVsRest Logistic Regression Baseline	0.866667	0.78	0.821053	50
KNN Baseline	0.844444	0.76	0.800000	50
OneVsRest SVM Baseline	0.823529	0.56	0.666667	50
SVM Baseline	0.904762	0.38	0.535211	50

Observations:

OneVsRest Random Forrest Gridsearch is my best model for recall, at a 92% and it's precision isn't bad at 90%, for the target class. However, I think I can increase performance by 1) coming back and retuning the hyper-parameters and by 2) combining that model with another model that has different strengths using ensemble methods or a voting classifier.

In [115]:
```
Image(filename = 'images/plots/OVR_RF_test_gridsearch_cm.png', width = 350)
```

executed in 20ms, finished 09:45:12 2022-07-06



My biggest concern for this model currently is about 13% of folks who are actually in the Suspected class, fail to be flagged as Suspected and miss out on further diagnostic tests. While 8% of Pathological diagnoses are misdiagnosed as Suspected by the model, I am less concerned about that 8 percent, as being diagnosed as Suspected should result in further testing and clinical support.

3.2 Exploring Feature Importances

3.3 Using SHAP For Feature Importances

In [116]:
```
shap.initjs()
```

executed in 21ms, finished 09:45:16 2022-07-06

In [117]:
```
rf_best_params = {'class_weight': 'balanced',
                  'criterion': 'entropy',
                  'max_features': 'auto',
                 'min_samples_leaf': 1,
                 'min_samples_split': 6}
```

executed in 7ms, finished 09:45:26 2022-07-06

In [118]:
```
shap_classifier = RandomForestClassifier(**rf_best_params)
```

executed in 5ms, finished 09:45:27 2022-07-06

In [119]:
```
shap_classifier.fit(X, y)
```

executed in 255ms, finished 09:45:27 2022-07-06

RandomForestClassifier

```
RandomForestClassifier(class_weight='balanced', criter
ion='entropy',
                       min_samples_split=6)
```

In [120]:
```
# Create Tree Explainer object that can calculate shap values
explainer = shap.TreeExplainer(shap_classifier)
```

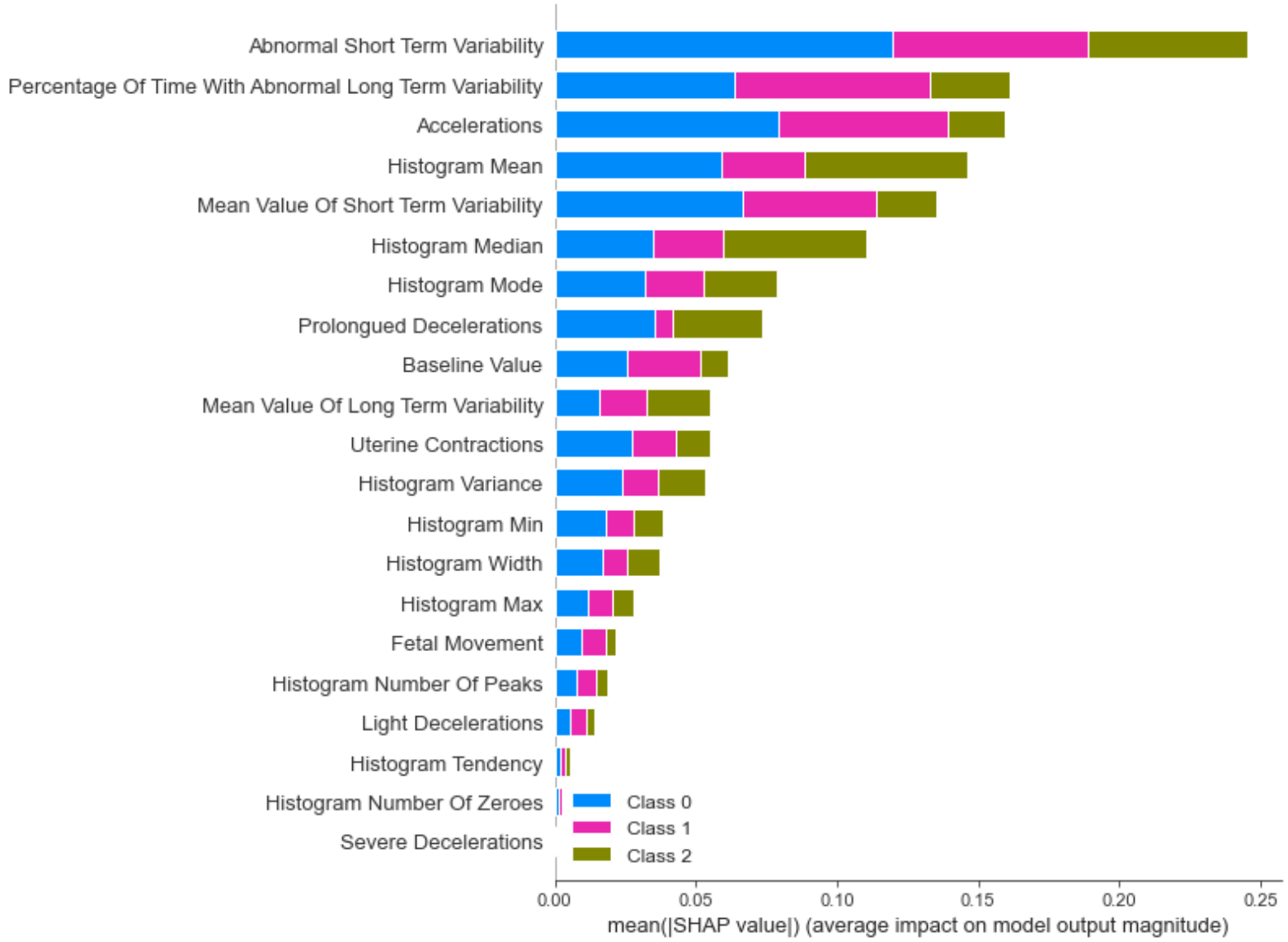
executed in 23ms, finished 09:45:28 2022-07-06

In [121]:
```
shap_values = explainer.shap_values(X_val, y_val)
```

executed in 1.44s, finished 09:45:30 2022-07-06

In [122]:
```
shap.summary_plot(shap_values, X_val, plot_type = 'bar', max_display = 40)
```

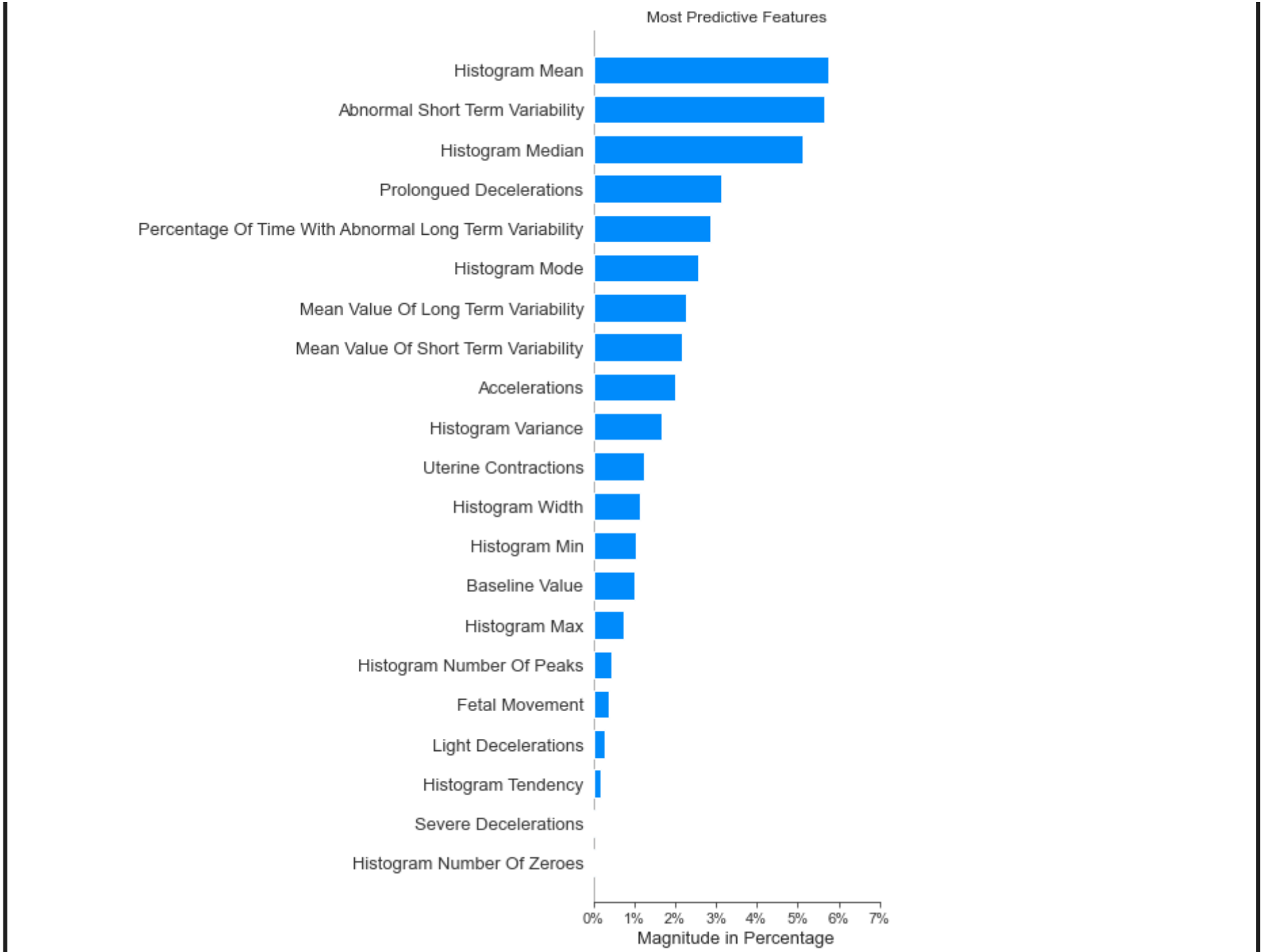
executed in 334ms, finished 09:45:31 2022-07-06



In [244]:

```
fig = shap.summary_plot(shap_values[2], X_val, plot_type = 'bar', max_display = 40,
                        title = 'Predictive Features', show = False)
plt.title("Most Predictive Features")
plt.xticks(ticks = [0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07],
           labels = ["0%", "1%", "2%", "3%", "4%", "5%", "6%", "7%"])
plt.xlabel("Magnitude in Percentage");

plt.tight_layout();
plt.savefig("SHAP Importances")
```

executed in 210ms, finished 11:56:37 2022-07-06


Most Predictive Features
Histogram Mean
Abnormal Short Term Variability
Histogram Median
Prolongued Decelerations
Percentage Of Time With Abnormal Long Term Variability
Histogram Mode
Mean Value Of Long Term Variability
Mean Value Of Short Term Variability
Accelerations
Histogram Variance
Uterine Contractions
Histogram Width
Histogram Min
Baseline Value
Histogram Max
Histogram Number Of Peaks
Fetal Movement
Light Decelerations
Histogram Tendency
Severe Decelerations
Histogram Number Of Zeroes
0%
1%
2%
3%
4%
5%
6%
7%
Magnitude in Percentage

Observations: Removing the columns with values less than Light Decelerations may reduce the dimensionality.

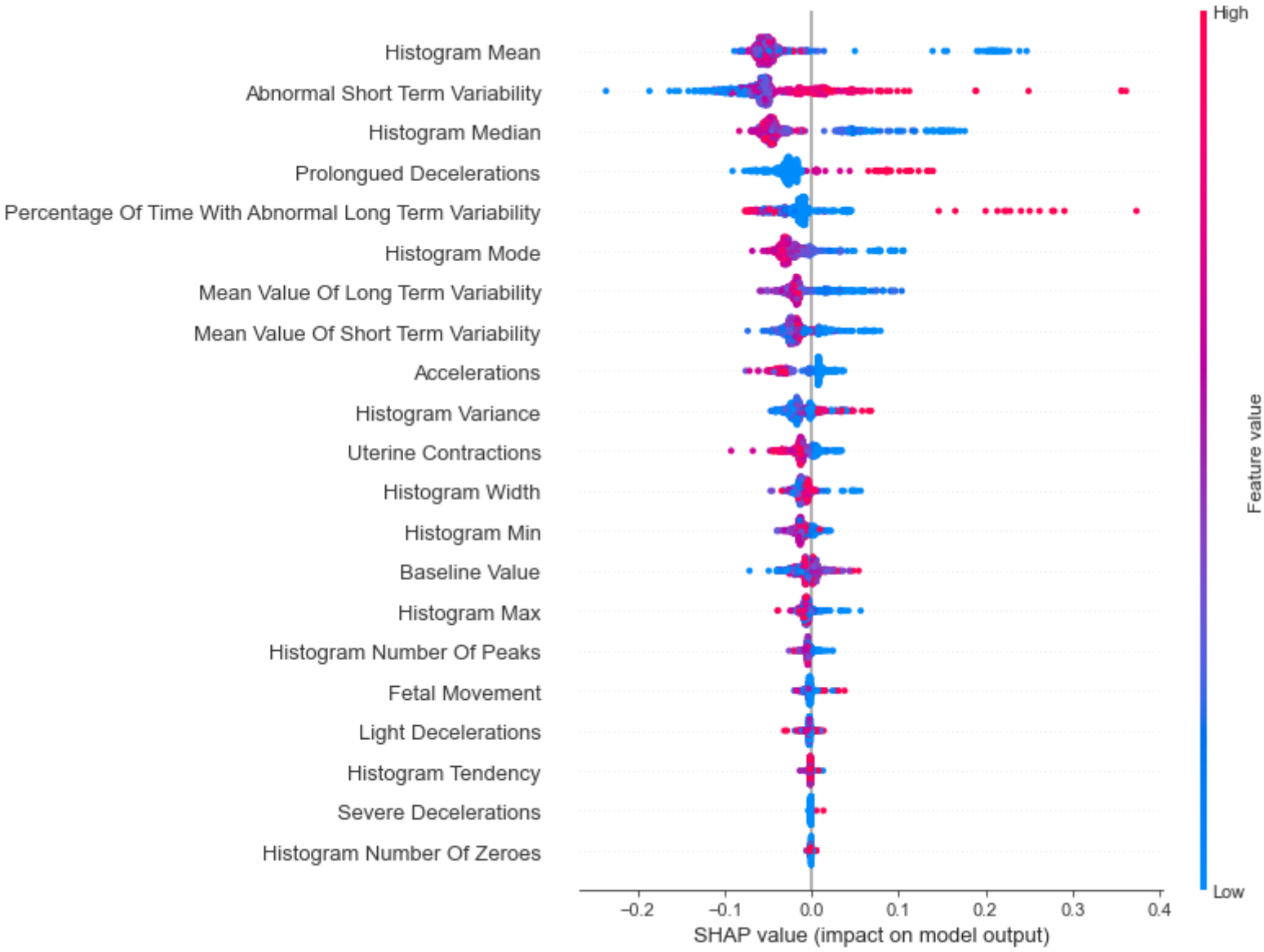
In [126]:

```
features_to_remove = ["Light Decelerations", "Histogram Tendency",
                      "Fetal Movement", "Histogram Number Of Zeroes",
                     "Severe Decelerations"]
```

executed in 3ms, finished 09:45:50 2022-07-06

In [127]:
```
shap.summary_plot(shap_values[2], X_val, plot_type = 'dot', max_display = 40)
```

executed in 394ms, finished 09:45:51 2022-07-06



Observations:

1) The lower the histogram median, the more likely someone is in the Pathological class.

2) The lower the short term variability, the less likely someone is in the Pathological class.

3) The higher the prolonged decelerations, the more likely someone is in the Pathological class.

3.3.1 Exploring Feature Importances to The Pathological Class Using SKLearn

In [128]:

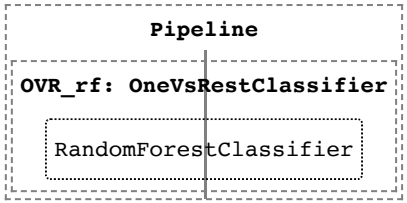
```
over_rf_best_params = {'OVR_rf__estimator__class_weight': 'balanced',
 'OVR_rf__estimator__criterion': 'entropy',
 'OVR_rf__estimator__max_features': 'auto',
 'OVR_rf__estimator__min_samples_leaf': 1,
 'OVR_rf__estimator__min_samples_split': 8}
```

executed in 6ms, finished 09:46:03 2022-07-06

In [129]:

```
OVR_rf_pipeline.set_params(**over_rf_best_params)

OVR_rf_pipeline.fit(X_train, y_train)
```

executed in 457ms, finished 09:46:04 2022-07-06



In [130]:

```
# grabbing only the feature importances from the Pathological class (estimators_[2])
feature_importance_vals = OVR_rf_pipeline[0].estimators_[2].feature_importances_

# putting those values into a dataframe and clean it up for graphing
feature_importance = pd.DataFrame(zip(title_labels, feature_importance_vals))
feature_importance.rename({0 : "Feature", 1: "Importance"}, axis = 1, inplace = True)
feature_importance.sort_values(by=['Importance'], ascending= False, inplace = True)
feature_importance
```

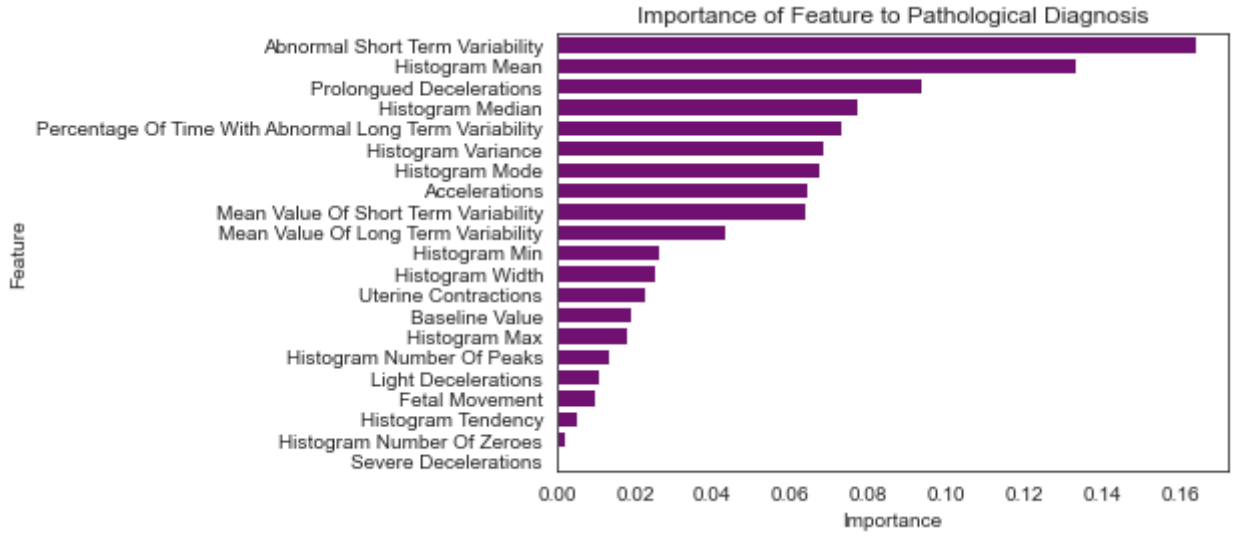
executed in 15ms, finished 09:46:04 2022-07-06

10	Mean Value Of Long Term Variability	4.510651e-02
12	Histogram Min	2.618302e-02
11	Histogram Width	2.523795e-02
3	Uterine Contractions	2.254355e-02
0	Baseline Value	1.923829e-02
13	Histogram Max	1.771995e-02
14	Histogram Number Of Peaks	1.344168e-02
4	Light Decelerations	1.098073e-02
2	Fetal Movement	9.931857e-03
20	Histogram Tendency	5.117647e-03
15	Histogram Number Of Zeroes	2.087932e-03
5	Severe Decelerations	2.297954e-18

In [131]:

```
feature_importance_plot = sns.barplot(data = feature_importance,
                                      y = "Feature",
                                      x = "Importance",
                                      color = "Purple",
                                      orient = "h")
feature_importance_plot.set_title("Importance of Feature to Pathological Diagnosis");
#plt.savefig("Importance of Feature to Pathological Diagnosis");
```

executed in 193ms, finished 09:46:07 2022-07-06



3.4 Final Model Dimension Reduction

In [189]:

```
rd_X_train = X_train.drop(columns = features_to_remove)
rd_X_val = X_val.drop(columns = features_to_remove)
rd_X_holdout = X_holdout.drop(columns = features_to_remove)
```

executed in 10ms, finished 10:10:19 2022-07-06

In [202]:

```
final_OVR_rf_pipeline = Pipeline([
    ('final_OVR_rf', OneVsRestClassifier(RandomForestClassifier(random_state = 42))),

])
baseline_params = {}

grid = GridSearchCV(final_OVR_rf_pipeline, baseline_params, scoring = 'recall_micro')
```

executed in 6ms, finished 10:28:56 2022-07-06

In [203]:

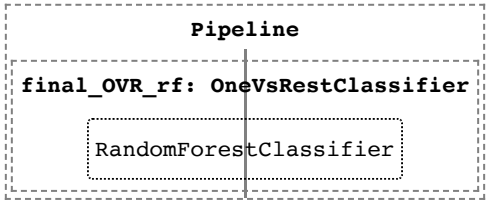
```
final_OVR_rf_gscv_params = {
    'final_OVR_rf__estimator__min_samples_split': [4, 5, 6, 7, 8, 9, 10],
    'final_OVR_rf__estimator__min_samples_leaf' : [1, 2, 3, 4, 5, 6],
    'final_OVR_rf__estimator__max_features' : ['auto', 'sqrt', 'log2'],
    'final_OVR_rf__estimator__criterion': ["gini", "entropy"],
    'final_OVR_rf__estimator__class_weight': ['balanced', y_weights]}

grid = GridSearchCV(final_OVR_rf_pipeline, final_OVR_rf_gscv_params,
                            scoring = 'recall_micro')
```

executed in 9ms, finished 10:28:57 2022-07-06

In [204]:

```
1 final_OVR_rf_pipeline.fit(rd_X_train, y_train)
```

executed in 487ms, finished 10:28:58 2022-07-06



In [206]:

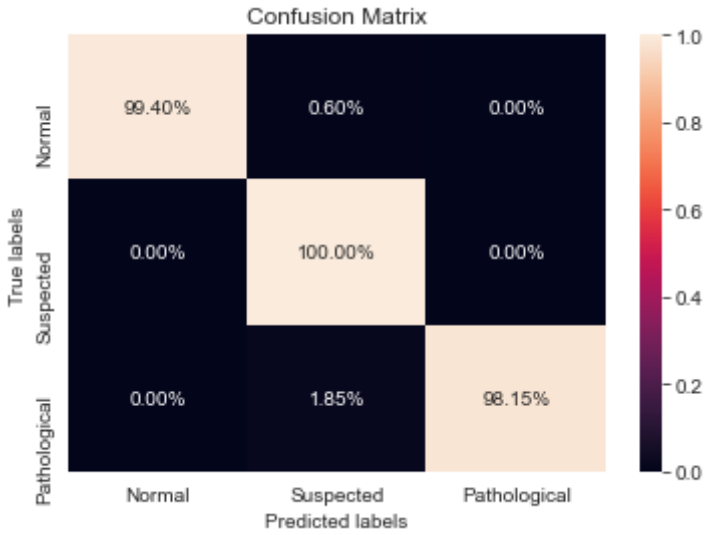
```
1 y_preds = final_OVR_rf_pipeline.predict(rd_X_holdout)
2 y_preds = lb.transform(y_preds)
3 #y_holdout = lb.transform(y_holdout)
```

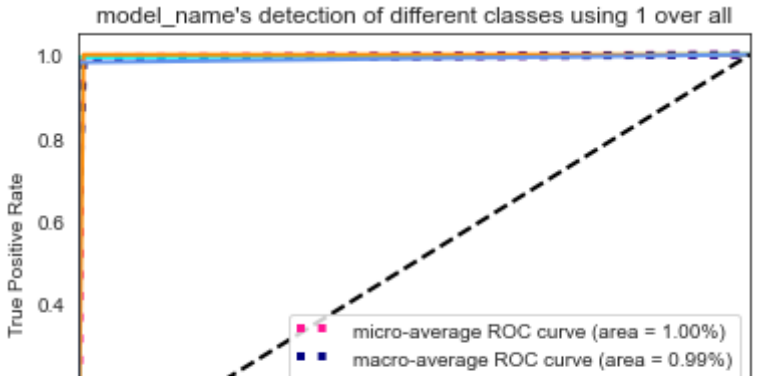
executed in 48ms, finished 10:29:08 2022-07-06

In [207]:
```
1 show_train_and_test(grid, rd_X_train, y_train, rd_X_val, 'Final Model')
```

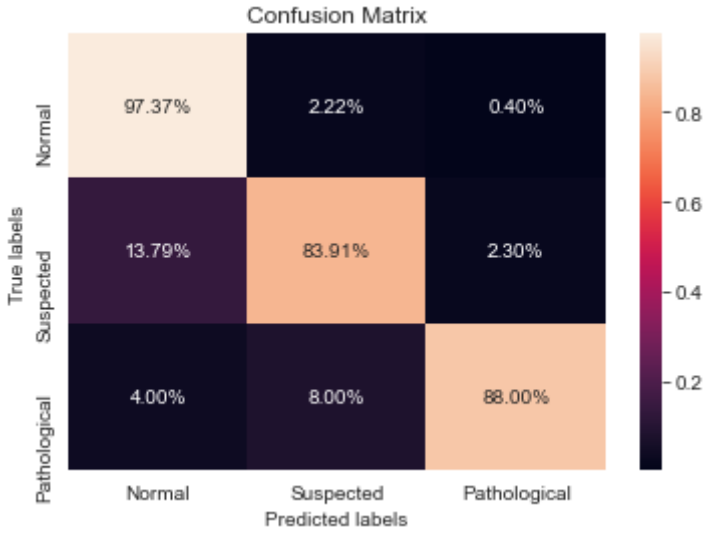
executed in 8m 6s, finished 10:37:17 2022-07-06

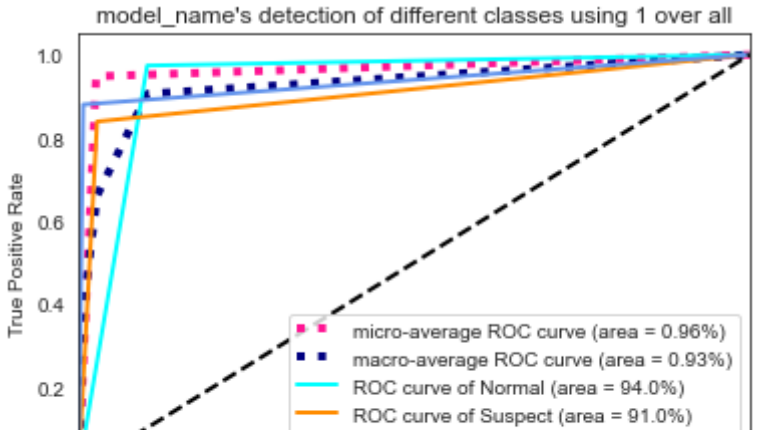
```
TRAINING SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
**********************************************************
[[0.99400599 0.00599401 0.        ]
 [0.         1.         0.        ]
 [0.         0.01851852 0.98148148]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 1.0, 'recall': 0.9814814814814815, 'f1-score': 0.9906542056074767, 'support': 108}
```






```
TEST SET CONFUSION MATRIX AND ROC AUC MULTICLASS PLOT
******************************************************
[[0.97373737 0.02222222 0.0040404 ]
 [0.13793103 0.83908046 0.02298851]
 [0.04       0.08       0.88      ]]
METRICS FOR PATHOLOGICAL CLASS
{'precision': 0.9166666666666666, 'recall': 0.88, 'f1-score': 0.8979591836734694, 'support': 50}
```






```
Best Parameters:
{'final_OVR_rf__estimator__class_weight': 'balanced', 'final_OVR_rf__estimator__criterion': 'gini', 'final_OVR_rf__esti
mator__max_features': 'auto', 'final_OVR_rf__estimator__min_samples_leaf': 2, 'final_OVR_rf__estimator__min_samples_spl
it': 5}

**********************************************************
```

3.4.1 Observations:

This model looks to be very overfit. In future work, I'd like to come back and better tune the hyper parameters to see if I couldn't get a model that generalizes much better.

In [208]:
```
target_names = ['Normal', 'Suspected', 'Pathological']
print(classification_report(y_holdout, y_preds, target_names=target_names))
```

executed in 24ms, finished 10:42:21 2022-07-06

```
                precision    recall  f1-score   support

        Normal       0.97      0.96      0.97       159
     Suspected       0.84      0.89      0.86        36
  Pathological       0.95      1.00      0.97        18

     micro avg       0.95      0.95      0.95       213
     macro avg       0.92      0.95      0.93       213
  weighted avg       0.95      0.95      0.95       213
   samples avg       0.95      0.95      0.95       213
```

Observations:

I made the holdout set too small, and would like to speak with the data collection team to discuss testing more data, for more conclusive results. However, the results for this very small sample are positive, and did better than the test set in the target class, with a 94% precision and recall score for the Pathological class.

The biggest problem with using this model as is, is it's performance in the Suspected class, where the model isn't as trustworthy nor as sensitive.

In [201]:
```
fetal_health_voting_model = 'final_model.sav'
pickle.dump(final_model, open(fetal_health_voting_model, 'wb'))
```

executed in 23ms, finished 21:33:04 2022-07-05

4 CONCLUSION

4.1 Future Work:

1) I'd like to revisit these models, especially my final model to retune them and see if I can't improve performance. Some of my models may have improved recall for the target class, or be more well rounded than my final model if they were less overfit.

2) I'd like to speak with the data collection team and see if we could negotiate a new class or indicator in the data to mark lethal prenatel diagnoses. Only a fraction of the pathological diagnoses our current data represents are cases of lethal prenatal diagnoses.

3) I'd like to iteratively test how I can reduce the dimensionality without sacrificing performance. Reducing the amount of tests needed to run for this model to produce reliable results will help the model's utility in the real world.

3) Most importantly: I want to make a secondary model. Right now, about 16% of Suspected cases were misclassified as Normal by my final model. I would like to make another model more sensitive to the "Suspected" class and have someone's metrics run through both models as a safety for that 16% of Suspected cases.

4.2 Recommendations: ¶

Birthing centers, hospitals and women's health centers should be on the look out for some early indicators:

1) A low histogram median.

2) A low short term variability.

3) A high prolonged decelerations.

These signs all point to fast tracking that patient for further diagnostic testing.

These are the simplest-to-interpret, most expressive features in my research targeting Pathological Diagnoses.

For data collection:

1) We can reduce the amount of data we are collecting for the purposes of this model, as proven by the model with reduced dimensions.

2) We can be more specific in our data collection towards lethal prenatel diagnoses specifically.

For Stacey:

1) Find a care team you trust to determine how many weeks you are in your pregnancy. This will affect the options in your area.

2) Get CTG Test and talk to your care team about documenting your preexisting condition.

3) Prescreen for pathological prenatal diganoses.

For Alex:

1) Setting up continuity of care between your natal team and your post natal team will be the priority.

2) Get CTG Test and talk to your care team about prescreening for pathological prenatal diagnoses.

3) Planning for infant mortality: funeral arrangements, etc.

4.3 Lastly:

This final model should be implemented by birthing centers, hospitals and women's health clinics as an early screening detection tool towards reducing Maternal Mortality rates, giving birthing parents a quicker diagnosis, and the best possible outcome in accordance to their own wishes.