CSE 422

ARTIFICIAL INTELLIGENCE

A PROJECT ON

FACE MASK DETECTION

GROUP 07

Maliha Mussarrat (20101354)

Rakibul Hassan Hredoy (20101357)

EftaKhairul Islam (20101390)

Md. Nazmus Sakib (20101613)

Introduction

Artificial Intelligence and Neural networks have already transformed internet technologies across the globe from something useful to something significant in our lives. Artificial intelligence innovations have intervened in all fields ranging from improved health care, where these machines are better suited to the detection of any diseases than any doctor and so on. At the end of 2019, the world was confronted with a widespread problem in the history of humanity: the Coronavirus pandemic (COVID-19). The COVID-19 virus affects the respiratory system and spreads through close contact. According to the World Health Organization, one of the most effective protective measures against the COVID-19 virus is to wear a face mask when in public areas.

Nevertheless, the report presented in our baseline paper as well as a few other papers hold promising results for accurately detecting the wear of face masks on individuals. The goal of this paper is to create a model that accurately determines whether an individual is wearing a mask or not wearing a mask. In this paper, we aimed to explore and test the new detection model and evaluation algorithms and their efficacy on masked faces. We will also used kaggle datasets, to test how our models perform for different groups of people. This project will be done using Python 3.8, NumPy, pandas, sklearn, matplotlib, os, cv2, seaborn, etc. The paper chosen for this project will be used as a baseline for this project. The ultimate aim of this project is to learn.

Methodology

Dataset Description: We'll use a Kaggle dataset of with mask (class label "1") vs without mask (class label "0") image comparisons, which has the images already divided into with mask and without mask folders. The data set consists of 7553 RGB images in 2 folders as with mask *and without mask*. Images of faces with masks are 3725 and images of faces without masks are 3828.

Libraries used: The project would use the following libraries:

- *Pandas* - for reading the data file
- *Numpy* - for arrays
- *Sklearn* - for splitting the dataset
- *Matplotlib* - for plotting graphs
- *OS* – used for creating and modifying directory
- *CV2* – for reading data from an image
- *Seaborn* – helps in exploration and understanding of data

Project Architecture: We used a Decision Tree classifier, Logistic Regression, and Naive Bayers to train our model using the Kaggle dataset. We started our preprocessing before dividing our data into training and testing datasets. After splitting, we compiled and fitted the model to the training dataset to build our model. We also extracted the model after compilation and fitting. We used the model to run our test set and got the results or outcomes we were looking for.

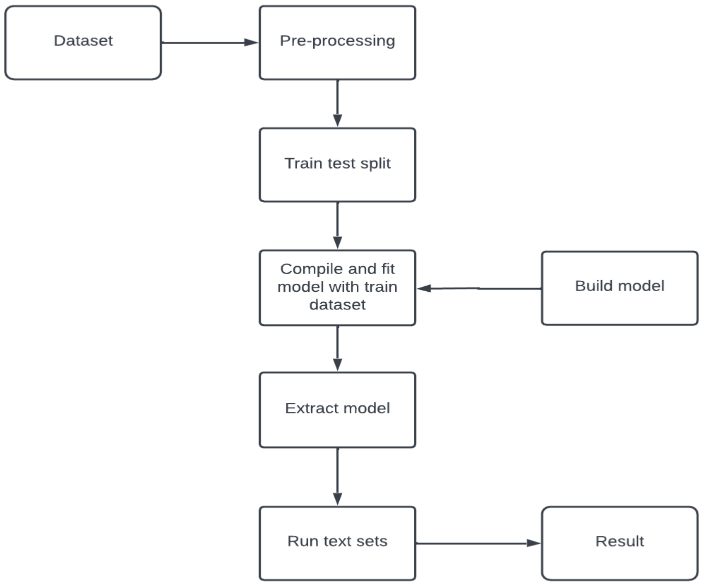
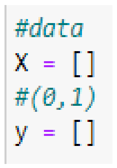


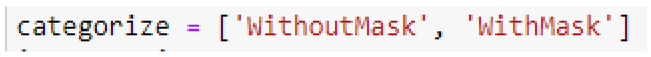
Figure: Project Architecture

Data Pre-processing: For processing our dataset, first of all, we created 2 arrays and used a dictionary for categorical arrays.



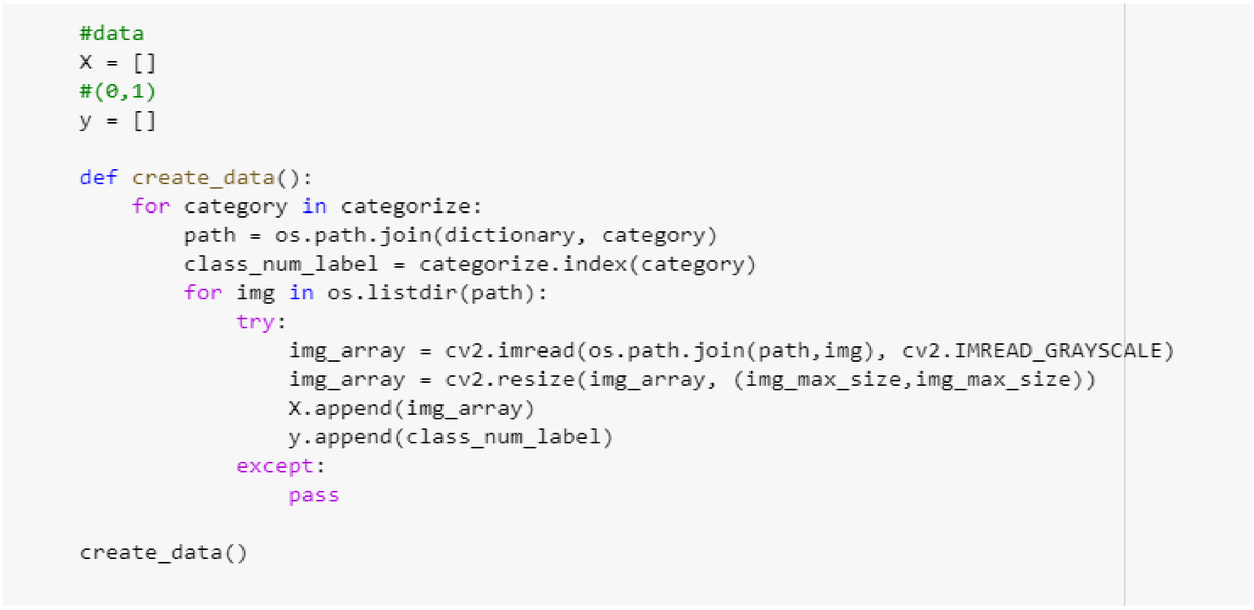
```
#data
X = []
#(0,1)
y = []
```

Declaring arrays



```
categorize = ['WithoutMask', 'WithMask']
```

Dictionary for categorical arrays

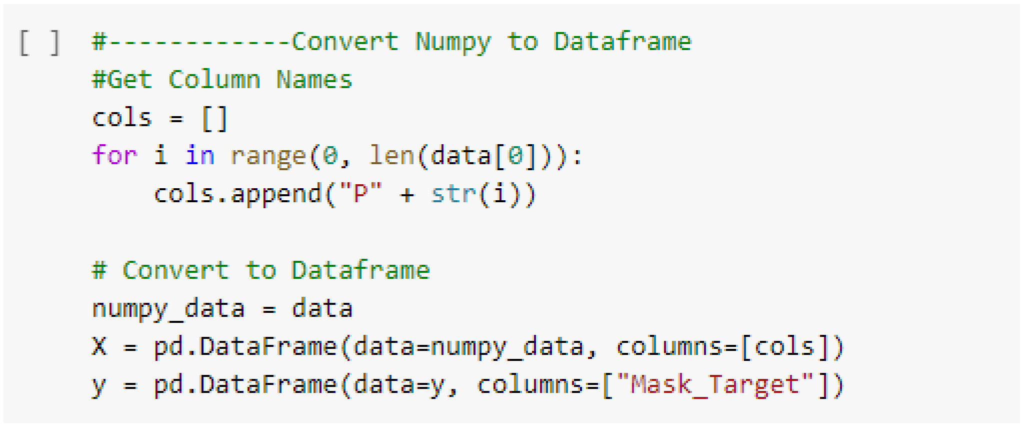


```python
#data
X = []
#(0,1)
y = []

def create_data():
    for category in categorize:
        path = os.path.join(dictionary, category)
        class_num_label = categorize.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img), cv2.IMREAD_GRAYSCALE)
                img_array = cv2.resize(img_array, (img_max_size,img_max_size))
                X.append(img_array)
                y.append(class_num_label)
            except:
                pass

create_data()
```

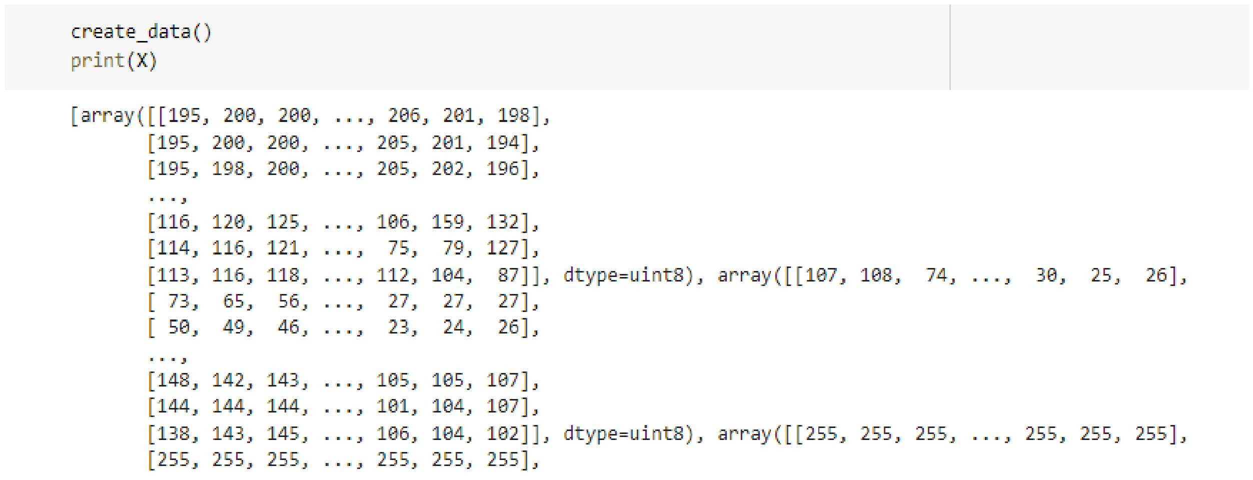
Data pre-processing



```python
[ ]  #------------Convert Numpy to Dataframe
     #Get Column Names
     cols = []
     for i in range(0, len(data[0])):
         cols.append("P" + str(i))

     # Convert to Dataframe
     numpy_data = data
     X = pd.DataFrame(data=numpy_data, columns=[cols])
     y = pd.DataFrame(data=y, columns=["Mask_Target"])
```

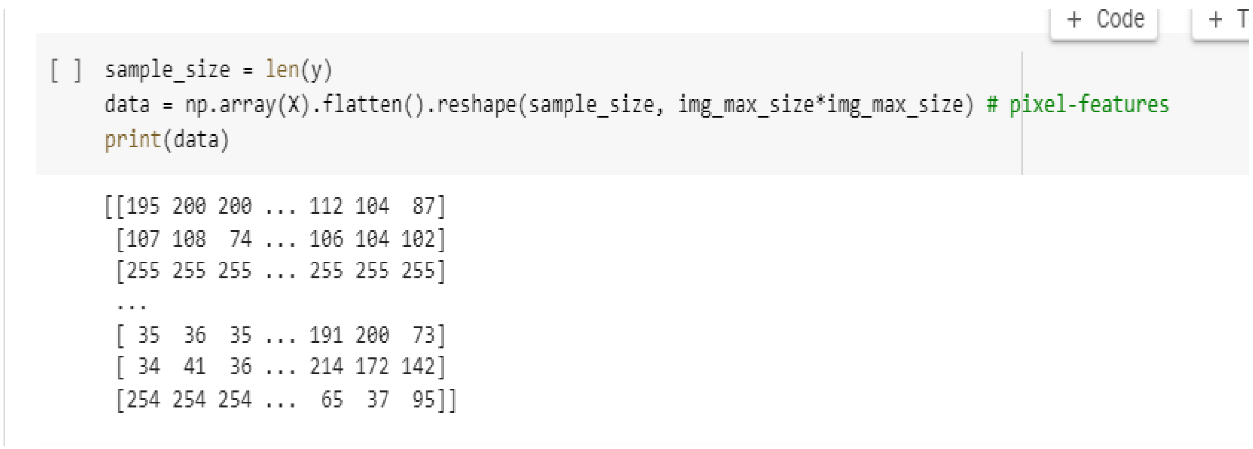
Changing Numpy array to Dataframe



```
create_data()
print(X)
```

```
[array([[195, 200, 200, ..., 206, 201, 198],
       [195, 200, 200, ..., 205, 201, 194],
       [195, 198, 200, ..., 205, 202, 196],
       ...,
       [116, 120, 125, ..., 106, 159, 132],
       [114, 116, 121, ...,  75,  79, 127],
       [113, 116, 118, ..., 112, 104,  87]], dtype=uint8), array([[107, 108,  74, ...,  30,  25,  26],
       [ 73,  65,  56, ...,  27,  27,  27],
       [ 50,  49,  46, ...,  23,  24,  26],
       ...,
       [148, 142, 143, ..., 105, 105, 107],
       [144, 144, 144, ..., 101, 104, 107],
       [138, 143, 145, ..., 106, 104, 102]], dtype=uint8), array([[255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
```

Actual Data

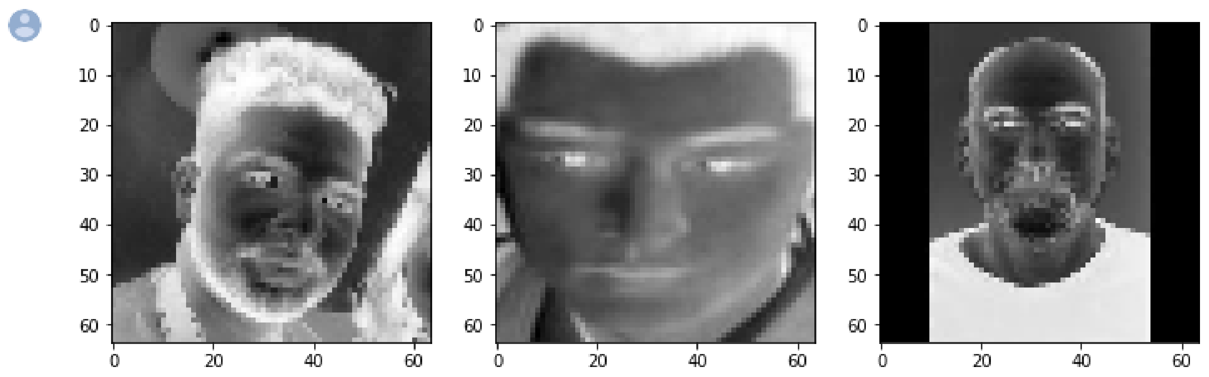


```
[ ] sample_size = len(y)
    data = np.array(X).flatten().reshape(sample_size, img_max_size*img_max_size) # pixel-features
    print(data)
```

```
[[195 200 200 ... 112 104  87]
 [107 108  74 ... 106 104 102]
 [255 255 255 ... 255 255 255]
 ...
 [ 35  36  35 ... 191 200  73]
 [ 34  41  36 ... 214 172 142]
 [254 254 254 ...  65  37  95]]
```

Data After Reshaping

Pre-Processing Techniques applied: In order to train our models with our dataset, we first had to normalize all the images before feeding them into our models. In order to "normalize" our photographs, each one had to be formatted in either grayscale or RGB (but not both), and each one had to have a uniform shape. For our project, we made the decision to downsize the photographs to 64X64 and format them in grayscale. We reasoned that since color had no bearing on recognizing masks, which vary in color, the additional information provided by an RGB image would not be required. Grayscale is a good choice because processing a grayscale image takes three to four times as less time as processing an RGB image. We chose a 64X64 image size because we believed it would not complicate our runtime and would still be readable. A sample of some of our pre-processed pictures is seen in the picture below.



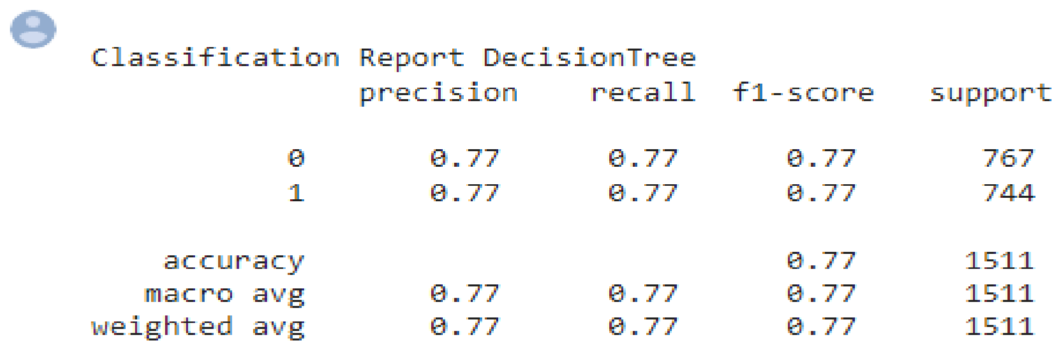
Pre-processed picture

Models applied: The models we applied in our projects are:

- Naive Bayes
- Logistic Regression
- Decision Tree Classifier

Rather than these models, we have used some of the classifier models such as SVC and KNN for our experiment. But we have not given any reference or report for these models in our report.

Decision Tree Classifier: Our report introduces the Decision Trees classifier as a new non-parametric model, and can be directly evaluated with the performance metrics for the Decision Trees model presented in our report. Although, it should be highlighted that differences in the training datasets are predicted to lead to inconsistencies in the predictions. The pixels were separated with a test size of 20% and a random state of 1, then normalized to a range of 0-1 from a range of 0-255 before the model was trained. The decision tree model's initial iteration used default settings for a single DecisionTreeClassifier() provided by scikit-learn. Then, we create a DecisionTreeClassifier model. We then test our testing data by fitting the training data and its accompanying labels. Moreover, due to the way the Decision Tree Classifier is structured as an algorithm, it works well with photos compared to other algorithms in our report. The Decision Tree reached only ~76% accuracy, ~77% recall, and ~77% f1 score. In conclusion, the Decision Tree Classifier performs well with images as opposed to the other models according to our report.



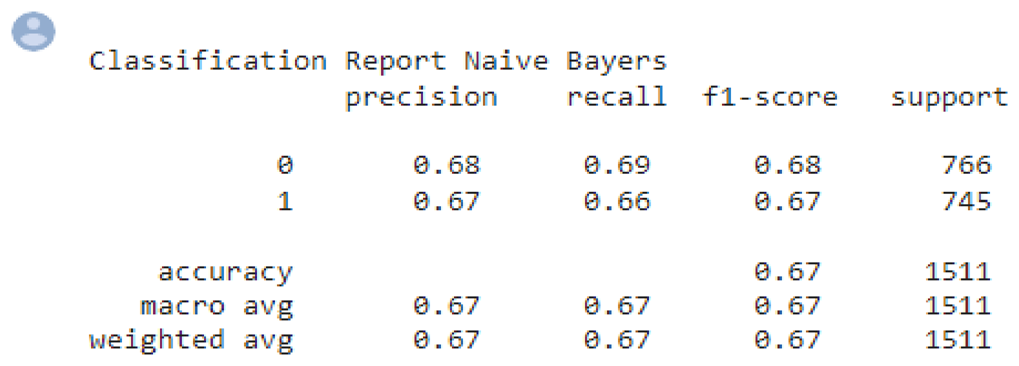
```
Classification Report DecisionTree
              precision    recall  f1-score   support

           0       0.77      0.77      0.77       767
           1       0.77      0.77      0.77       744

    accuracy                           0.77      1511
   macro avg       0.77      0.77      0.77      1511
weighted avg       0.77      0.77      0.77      1511
```

Classification Report of DecisionTree

Naive Bayes: We also included the Naive Bayes method in our project. We choose to include this algorithm since it is a powerful algorithm that is still used to identify spam in our emails. Because the naive Bayes algorithm makes the assumption that all features are conditionally independent, we predicted before developing this model that this kind of algorithm would not work well. The majority of pixels in any image are associated with one another when it comes to image classification; nonetheless, since the naive Bayes approach performs fairly well in natural language processing even when its assumption is false, we choose to test it out. We used sklearn's Gaussian Naive Bayes. After normalizing the dataset, we split 20% of the data for testing and the rest for training with a random state of 0. Then, we create a GaussianNB model. We then test our testing data by fitting the training data and its accompanying labels. Moreover, due to the way naive Bayes is structured as an algorithm, it doesn't work well with photos. The GaussianNB reached only ~67% accuracy, ~69% recall, and ~68% f1 score. In conclusion, Naive Bayes doesn't perform well with images because of its structure as an algorithm.



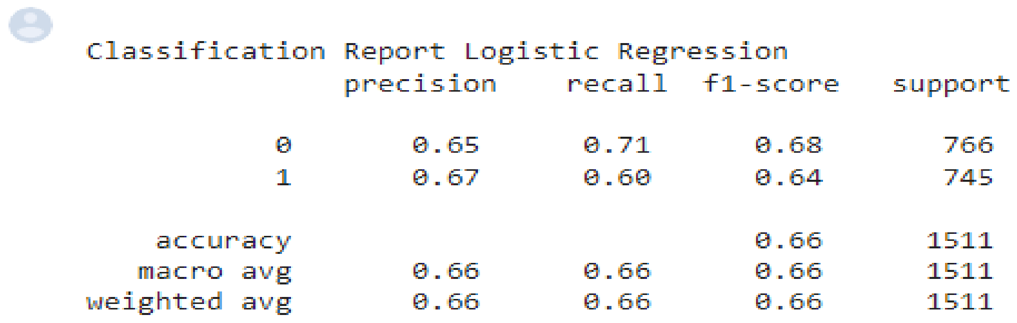
```
Classification Report Naive Bayers
              precision    recall  f1-score   support

           0       0.68      0.69      0.68       766
           1       0.67      0.66      0.67       745

    accuracy                           0.67      1511
   macro avg       0.67      0.67      0.67      1511
weighted avg       0.67      0.67      0.67      1511
```

.Classification Report of Naive Bayes

Logistic Regression: One of the most well-known machine learning algorithms that fall within supervised learning techniques is logistic regression. It can be applied to Classification and Regression issues but is primarily utilized for Classification issues. The output of the Logistic Regression problem can be only between 0 and 1. Due to the binary answer of the conditional dependent variable, where a face is either in the state of "with mask" or "without mask," logistic regression was used as the regression method for this study. The objective of this analysis, like with any regression model, is to build the most accurate, economical, and operationally comprehensible model to depict the connection between the dependent and independent variables. In our case of the logistic regression problem, 1 is used for with a mask, and 0 is for without a mask. We used sklearn's Logistic Regression Classifier . After pre-processing the dataset, we split 20% of the data for testing and the rest for training with a random state of 0. Then, we create a LogisticRegression model. We then test our testing data by fitting the training data into the model and its accompanying labels. Additionally, because of the algorithmic structure of the Logistic Regression, it performs poorly with images. Logistic Regression Classifier reached only ~65% accuracy, ~71% recall, and ~68% f1 score. In conclusion, Logistic Regression doesn't perform well with images because of its structure as an algorithm.



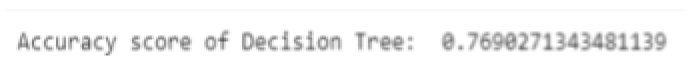
```
Classification Report Logistic Regression
              precision    recall  f1-score   support

           0       0.65      0.71      0.68       766
           1       0.67      0.60      0.64       745

    accuracy                           0.66      1511
   macro avg       0.66      0.66      0.66      1511
weighted avg       0.66      0.66      0.66      1511
```

Classification Report of Logistic Regression

Results: To sum up, our Decision Trees model performed the best with a testing accuracy of ~76%. Following is our Gaussian Naive Bayes with 67% accuracy and Logistic Regression with ~ 65% accuracy. The best classification overall for a single class category goes to the without face mask category which consistently performed with the highest accuracies compared to the other two classes.



```
Accuracy score of Decision Tree:  0.7690271343481139
```

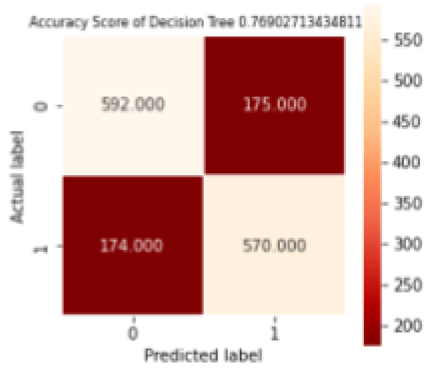
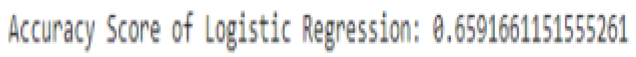


Figure: Accuracy Score & Heatmap of Decision Tree



```
Accuracy Score of Logistic Regression: 0.6591661151555261
```

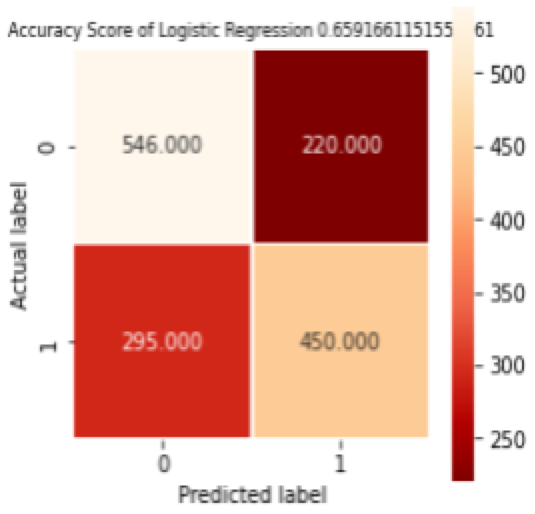
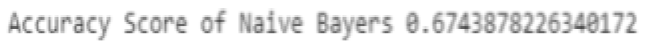


Figure: Accuracy Score & Heatmap of Logistic Regression



```
Accuracy Score of Naive Bayers 0.6743878226340172
```

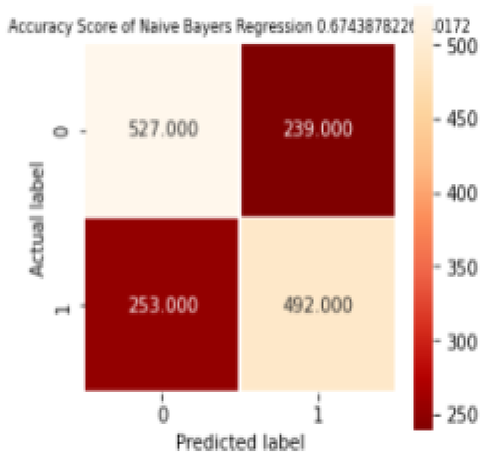


Figure: Accuracy Score & Heatmap of Naive Bayes

Conclusion

The significance of dimensionality reduction and computing expense was underlined through experiments with the non-parametric model for the decision trees. By eliminating irrelevant characteristics, we can potentially raise the accuracy of our model while simultaneously saving runtime by minimizing unneeded features. We also discovered the significance of dataset selection and data preprocessing. It is possible that inconsistencies in our datasets and image quality, such as zoomed-in photographs and simulated or Photoshopped face masks on faces, caused mistakes in the training of our models and hampered their ability to forecast unseen cases that varied in these respects. Overall, we learned the value of features in creating a successful classifier, how simple it is to overfit and underfit models, how to prevent doing so, and how data influences model biases. Despite the fact that some of our models performed well, such as Naive Bayes and Decision Tree, all of our models struggled to categorize newly added cases to our model. To increase our accuracy, more research on the shortcomings of our training datasets and the model setup is required. It's crucial to investigate any potential overfitting of our models to the training dataset. Before drawing any conclusions, though, it is important to further evaluate image perspective and image deformities in both datasets. Our model is unable to accurately forecast whether or not a mask will be used. It is something to consider while thinking about this study and when deciding on the best dataset to reflect the subtleties provided in proper mask wear.

References:

Omkar, Gurav, (2020). Face Mask Detection Dataset

Retrieved September 30, 2022, from

https://www.kaggle.com/datasets/omkargurav/face-mask-dataset?resource=download

Nithyashree, V, (January 20, 2022). Image Classification using Machine Learning

Retrieved September 30, 2022, from

https://www.analyticsvidhya.com/blog/2022/01/image-classification-using-machine-learn
ing/

Nikhil, Kumar, (23 August 2022). Understanding Logistic Regression

Retrieved September 30, 2022, from

https://www.geeksforgeeks.org/understanding-logistic-regression/