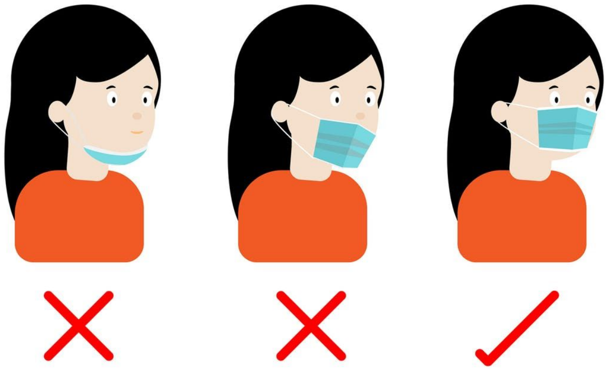
Face Mask Detection

Shania Dhani
Michelle Lucero
Xuejin Gao

Problem Description

Mask Restrictions in public areas & Contact Tracing for COVID-19



State of the Art/Related Work

“A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic” (Loey, Manogaran, Taha and Khalifa)

Does not account for *wearing masks incorrectly*

Has up to a 100% accuracy across one of the three tested datasets

Member	Role
Michelle	Data Preprocessing and SVM
Shania	KNN and Decision Tree Models
Xuejin	CNN and Naive Bayes Models

Data Preprocessing

KNN Progression

75%

No Hyperparameter Tuning

~15 mins

Reduced images to 64X64 pixels to reduce dimensionality in the dataset.

82%

GridSearchCV Hyperparameter Tuning Phase 1

Not Normalized

Manhattan, Neighbors: 5

~15 hours; Memory Intensive

Computationally Expensive for Predictions

~83%

Dimensionality Reduction Phase 2

Normalized Dataset

PCA transformation w/ 90% variance preservation over the training set

86.7%

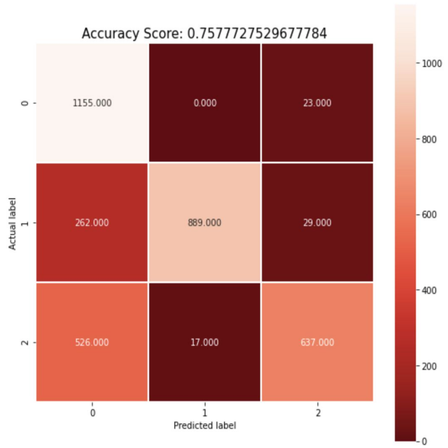
RandomizedSearchCV Hyperparameter Tuning Phase 3

Normalized Dataset

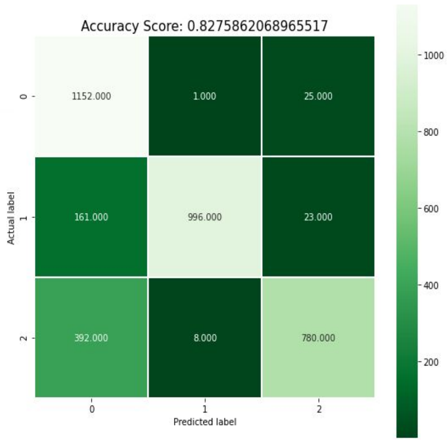
Manhattan, Neighbors: 2, weight: distance

Less computationally expensive + memory intensive

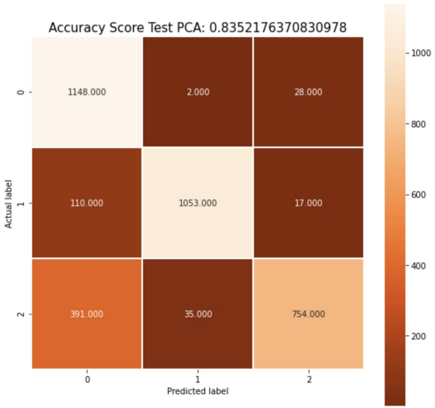
KNN Model Evaluation



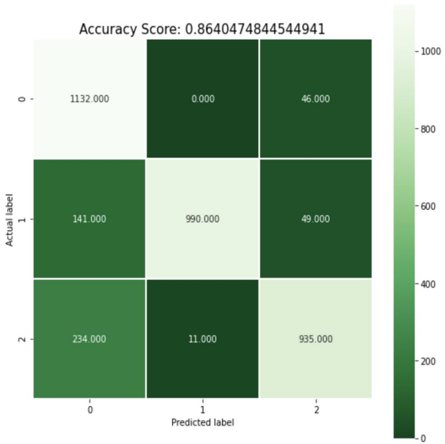
No Hyperparameter Tuning



GridSearchCV Hyperparameter Tuning



Principal Component Analysis



RandomizedSearchCV Hyperparameter Tuning

Decision Trees Progression

75%

83%

84%

No Hyperparameter Tuning

Fast

Default Scikit Parameters

Normalized

Tree is more shallow

Dimensionality Reduction: PCA

Fast

Transformation improved accuracy

Normalized

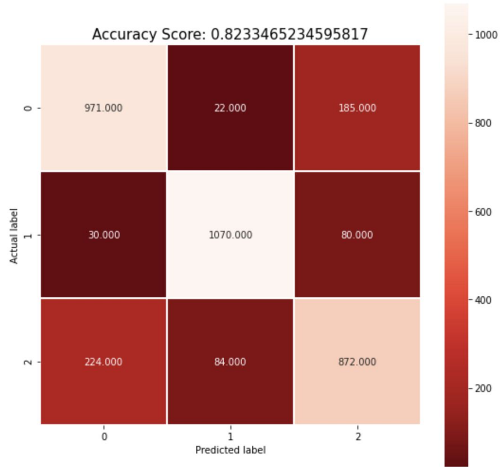
Hyperparameter Tuning

Fast

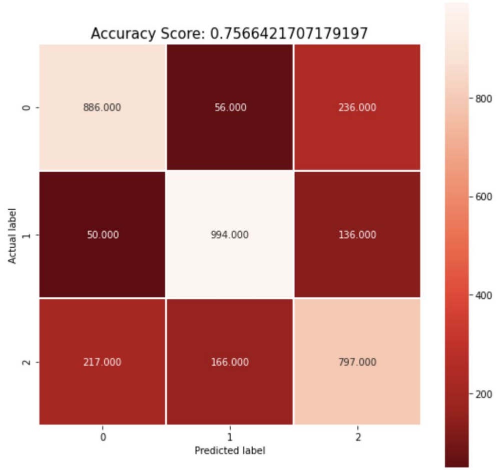
Gini, max depth = 10, max features = 1138

Searching for the model's best parameters was a ~1 hour computation

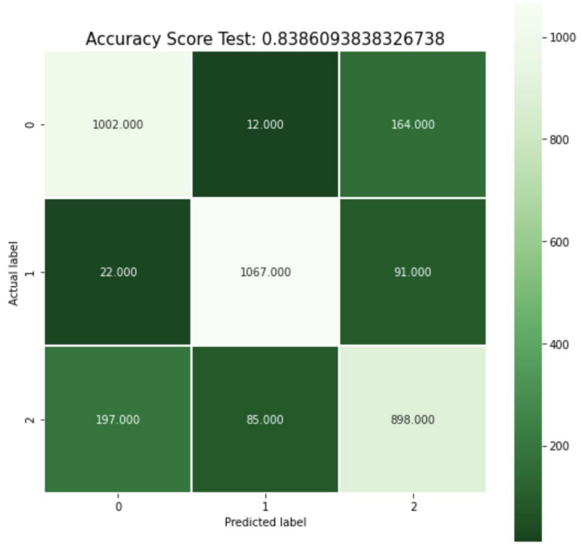
Decision Trees Model Evaluation



82%
No hyperparameter tuning



75%
Dimensionality Reduction
Of 90% variance



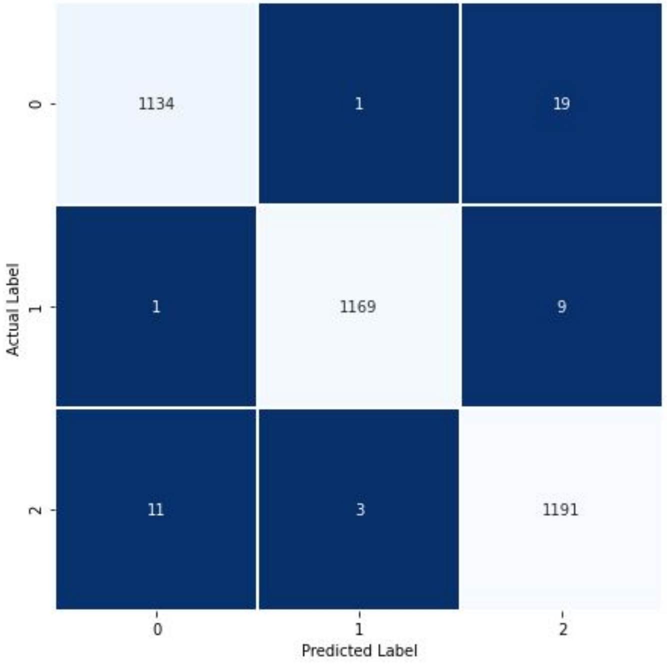
84%, Max-Depth=10, Max Features=1138

Hyperparameter Tuning Classification Report

	precision	recall	f1-score	support
0	0.82	0.85	0.84	1178
1	0.92	0.90	0.91	1180
2	0.78	0.76	0.77	1180
accuracy			0.84	3538
macro avg	0.84	0.84	0.84	3538
weighted avg	0.84	0.84	0.84	3538

CNN Metrics

	precision	recall	f1-score	support
0	0.99	0.98	0.99	1154
1	1.00	0.99	0.99	1179
2	0.98	0.99	0.98	1205
accuracy			0.99	3538
macro avg	0.99	0.99	0.99	3538
weighted avg	0.99	0.99	0.99	3538



SVM Metrics

Evaluate performance for using 100% of the dataset

```
### 1. Get and print a baseline accuracy score.
y_pred = model_100.predict(X_test)
accuracy = model_100.score(X_test, y_test)
print("Accuracy %f" % accuracy)
metrics.accuracy_score(y_true=y_test, y_pred=y_pred)
```

```
Accuracy 0.961556
```

```
5]: 0.9615558570782451
```

```
print(metrics.classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.95	0.96	0.96	1489
1	0.99	0.98	0.99	1466
2	0.94	0.94	0.94	1467
accuracy			0.96	4422
macro avg	0.96	0.96	0.96	4422
weighted avg	0.96	0.96	0.96	4422

Evaluate performance for using 75% of the dataset

```
### 1. Get and print a baseline accuracy score.
y_pred = model_75.predict(X_test)
accuracy = model_75.score(X_test, y_test)
print("Accuracy %f" % accuracy)
metrics.accuracy_score(y_true=y_test, y_pred=y_pred)
```

```
Accuracy 0.951161
```

```
9]: 0.9511606873681037
```

Evaluate performance for using 50% of the dataset

```
### 1. Get and print a baseline accuracy score.
y_pred = model_50.predict(X_test)
accuracy = model_50.score(X_test, y_test)
print("Accuracy %f" % accuracy)
metrics.accuracy_score(y_true=y_test, y_pred=y_pred)
```

```
Accuracy 0.939846
```

Evaluate performance for using 25% of the dataset

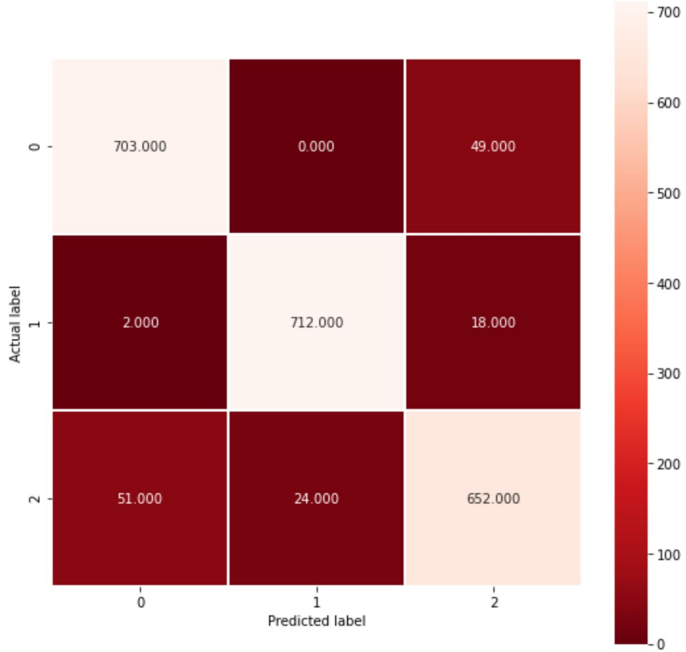
```
### 1. Get and print a baseline accuracy score.
y_pred = model_25.predict(X_test)
accuracy = model_25.score(X_test, y_test)
print("Accuracy %f" % accuracy)
metrics.accuracy_score(y_true=y_test, y_pred=y_pred)
```

```
Accuracy 0.933996
```

```
5]: 0.933996383363472
```

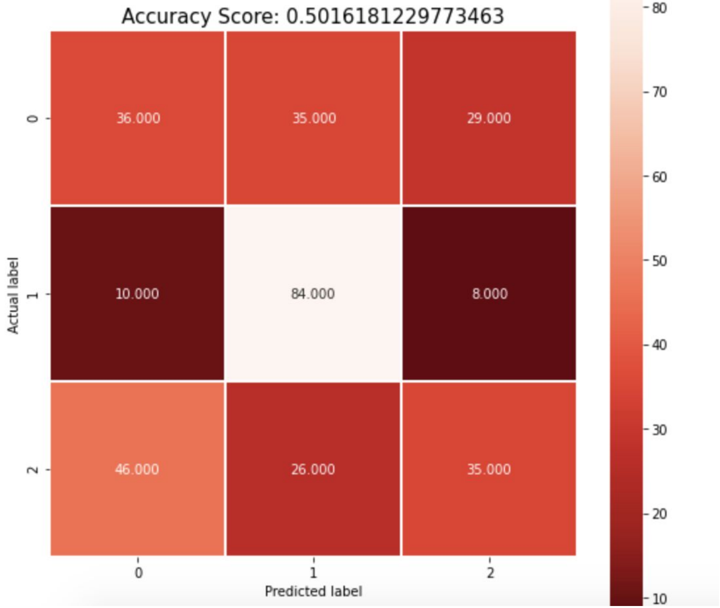
SVM Metrics

	precision	recall	f1-score	support
0	0.93	0.93	0.93	752
1	0.97	0.97	0.97	732
2	0.91	0.90	0.90	727
accuracy			0.93	2211
macro avg	0.93	0.93	0.93	2211
weighted avg	0.93	0.93	0.93	2211



KNN Model Bias Assessment

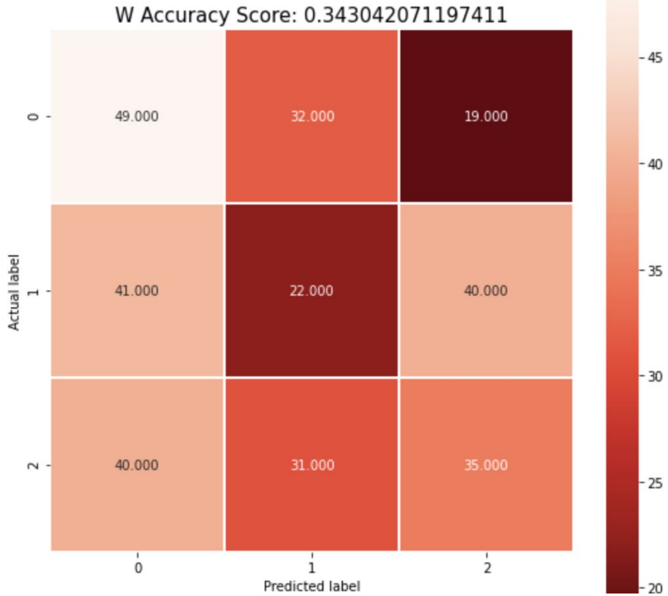
POC, ~50% testing accuracy



Classification Report POC Test Dataset

	precision	recall	f1-score	support
0	0.39	0.36	0.37	100
1	0.58	0.82	0.68	102
2	0.49	0.33	0.39	107
accuracy			0.50	309
macro avg	0.49	0.50	0.48	309
weighted avg	0.49	0.50	0.48	309

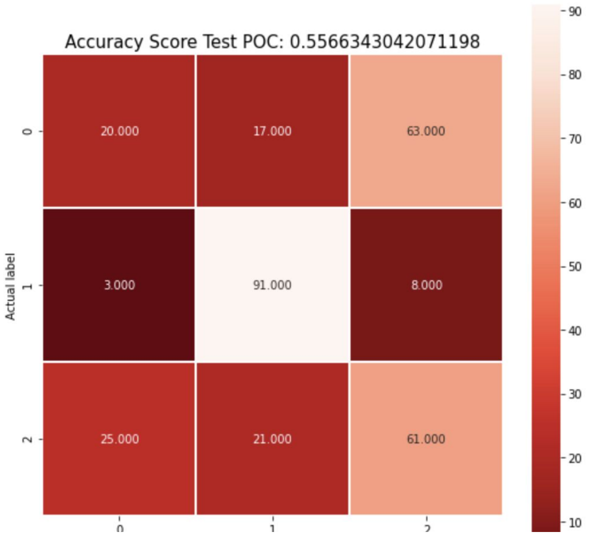
W, ~34% testing accuracy



Classification Report W Test Dataset

	precision	recall	f1-score	support
0	0.38	0.49	0.43	100
1	0.26	0.21	0.23	103
2	0.37	0.33	0.35	106
accuracy			0.34	309
macro avg	0.34	0.34	0.34	309
weighted avg	0.34	0.34	0.34	309

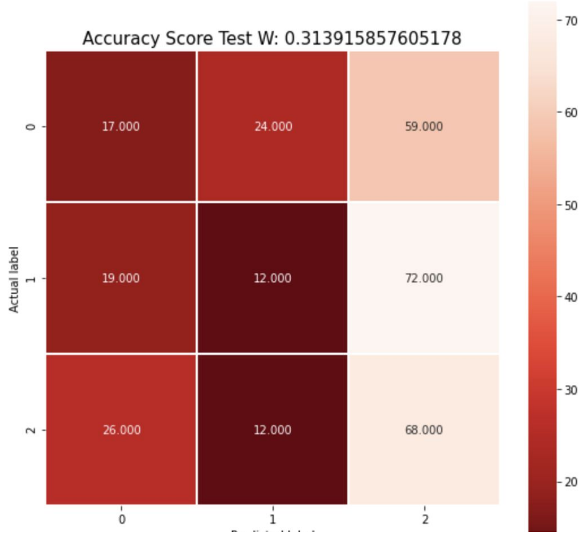
Decision Trees Bias Assessment



POC Classification Report

	precision	recall	f1-score	support
0	0.42	0.20	0.27	100
1	0.71	0.89	0.79	102
2	0.46	0.57	0.51	107
accuracy			0.56	309
macro avg	0.53	0.55	0.52	309
weighted avg	0.53	0.56	0.52	309

POC Dataset Accuracy 55.6%

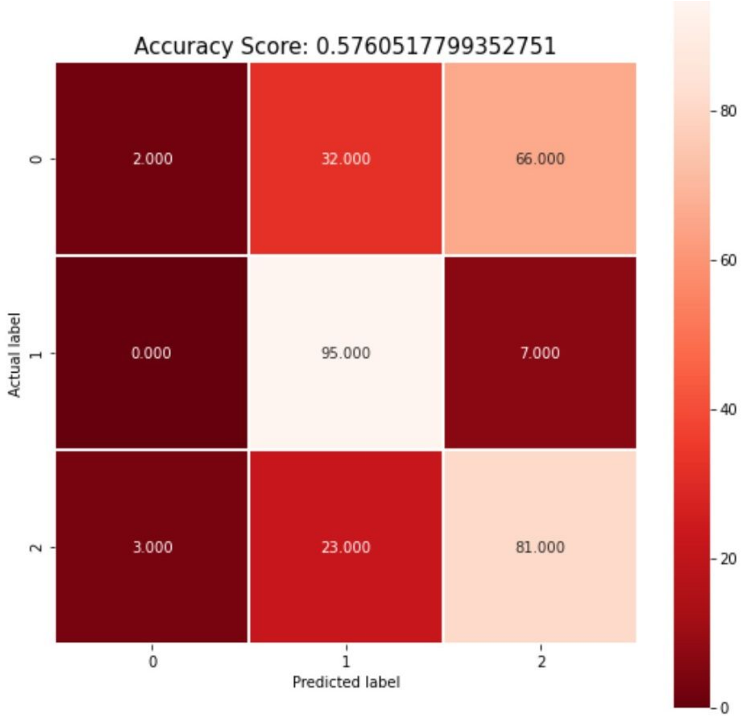


W Classification Report

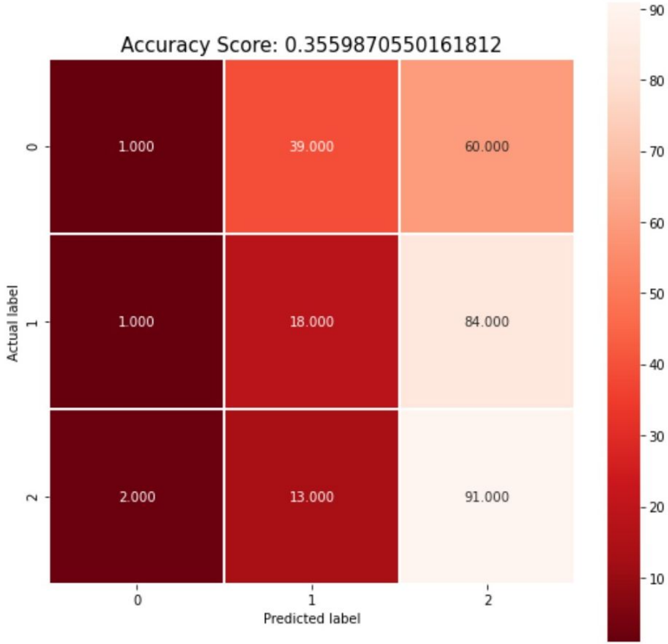
	precision	recall	f1-score	support
0	0.27	0.17	0.21	100
1	0.25	0.12	0.16	103
2	0.34	0.64	0.45	106
accuracy			0.31	309
macro avg	0.29	0.31	0.27	309
weighted avg	0.29	0.31	0.27	309

W Dataset Accuracy 31.4.%

SVM Bias Assessment



POC Dataset Accuracy 57.6%



W Dataset Accuracy 35.6%

CNN Bias Assessment

```
10/10 [==============================] - 0s 3ms/step - loss: 2.9011 - acc: 0.6052
POC dataset: [2.901106834411621, 0.6051779985427856]
10/10 [==============================] - 0s 3ms/step - loss: 4.4483 - acc: 0.3592
W dataset: [4.44830322265625, 0.35922330617904663]
```

POC: ~60% accuracy
W   : ~36% accuracy

Differences between Images

Image from our training data of class “incorrectly wearing mask”

Image from our testing W data of class “incorrectly wearing mask”

POSSIBLE REASONS BIAS ASSESSMENT PERFORMANCE:

- We overfitted our models to the training data
- The training images contain noise or features that differ dramatically from the new instances
- More Feature Selection was needed to reduce the importance of irrelevant features
- Simulated face masks in our training data, negatively affected our models ability to correctly classify certain types of instances

What We Learned

- Feature Set Is Important.
  - Too many “unimportant” features can be noisy,
  - Too many features can be computationally expensive and memory intensive for certain models (KNN, SVM)
- Dataset Collection and Choice is Important.
  - Be picky about your training dataset.
  - Actively test more often for biases in dataset
- Hyperparameter Tuning is Hard.

Flask App


Face-Mask Detection
Home
About Face-Mask Detection
Contributors
Face-Mask Image Classifier
Prediction for input image is: Correctly Worn
Correctly Wearing : 1.00
Incorrectly Wearing: 0.00
Not Wearing : 0.00


Face-Mask Detection
Home
About Face-Mask Detection
Contributors
Face-Mask Image Classifier
Prediction for input image is: Without A Mask
Correctly Wearing : 0.06
Incorrectly Wearing: 0.02
Not Wearing : 0.92

Thank You!

The End

References

Loey, M., Manogaran, G., Taha, M., & Khalifa, N. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement : journal of the International Measurement Confederation*, *167*, 108288. https://doi.org/10.1016/j.measurement.2020.108288

Evaluation

"99.64% accuracy, with up to 100% accuracy on one of the three different datasets used in the baseline research."

We will use the performance metrics gathered on the:

- SVM
- KNN
- CNN
- Naive Bayes
- Decision Trees

To make our own ML Model and compare it to our baseline study.

Approach

We plan to use a combination of different datasets available to create a new dataset that is labelled ***for mask v. no mask v. incorrect wearing of a mask.***

- KNN
- Different types of CNN to gauge which models are the best predictors.

We hope to create our own face-detection model.