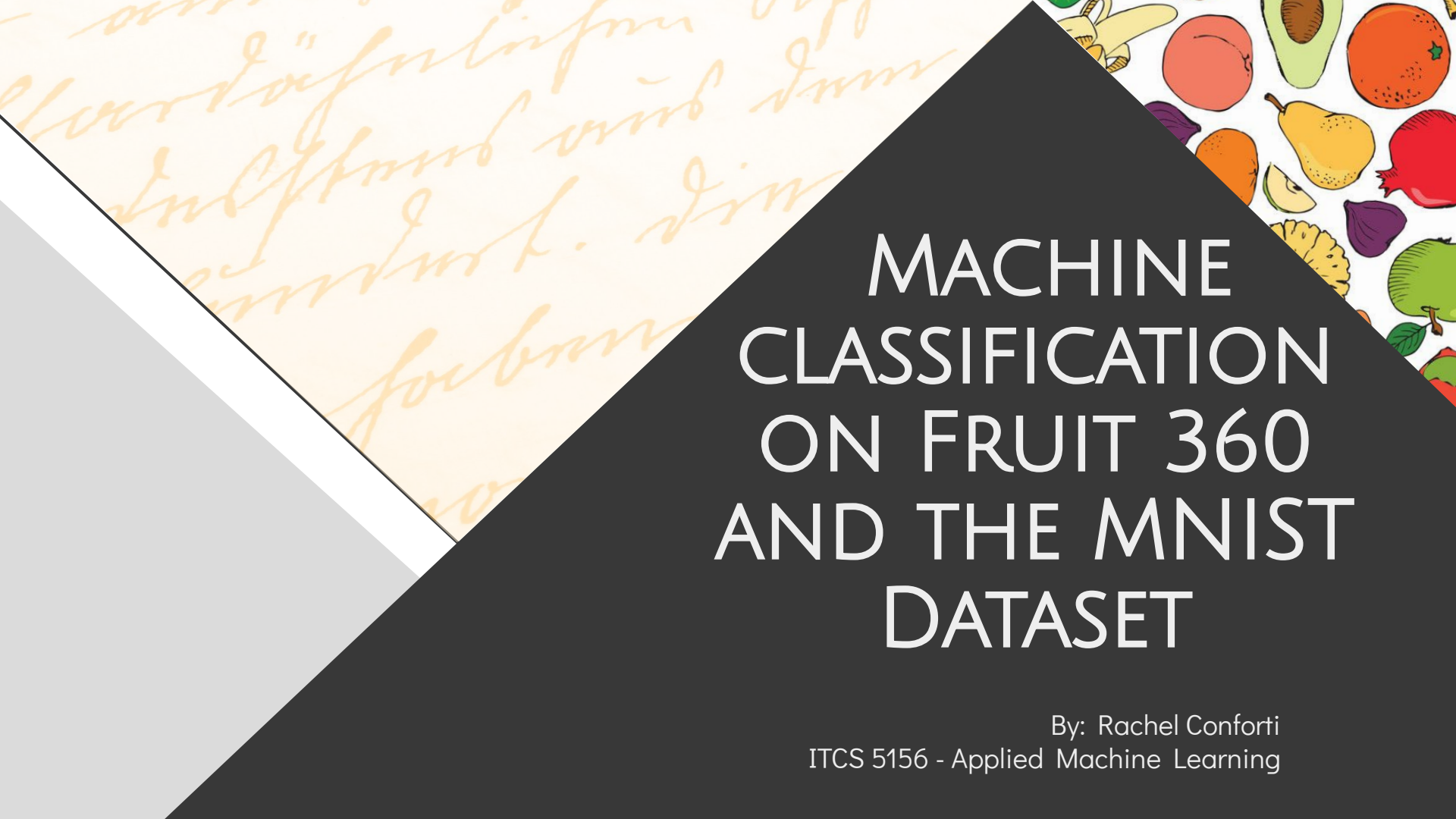


Machine Classification on Fruit 360 and the MNIST Dataset

By: Rachel Conforti
ITCS 5156 - Applied Machine Learning

Problem Statement

- The goal of my project was to take the Fruit 360 dataset, process it to have both a grayscale dataset and a binary grayscale dataset and see the performance based on these models: Support Vector Machine, Decision Tree, Naïve Bayes, K-Nearest Neighbor, and Random Forest.

- I will use the MNIST dataset which was used in paper and a dataset from Kaggle called Fruit 360, which is a dataset of various fruit!

Grayscale photo

Binary Grayscale photo

Motivation and Challenges

- Always been a personal interest of mine despite no previous experience on how to do this!
- This technology is being implemented all over the world as seen in self-driving cars, depositing checks by taking a simple photo of it, or even turning someone's handwritten notes into typed notes automatically.
- Wanted to try first hand what this would encompass to accomplish this task on something simple, such as handwriting and fruit!

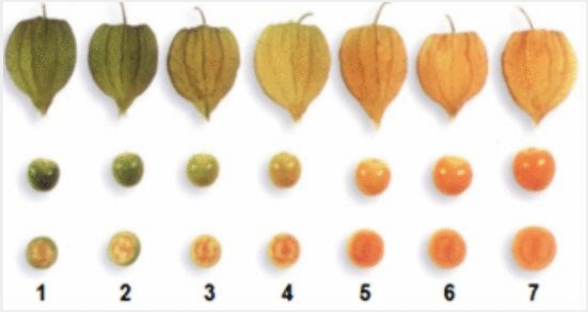
Challenges:

- Ensuring that the datasets are reading in as true grayscale and binary grayscale.
- Tweaking the preprocessed datasets

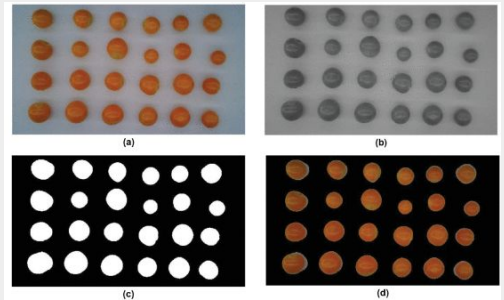
Related Approaches

Classification of Cape Gooseberry Fruit and its Different Color Spaces

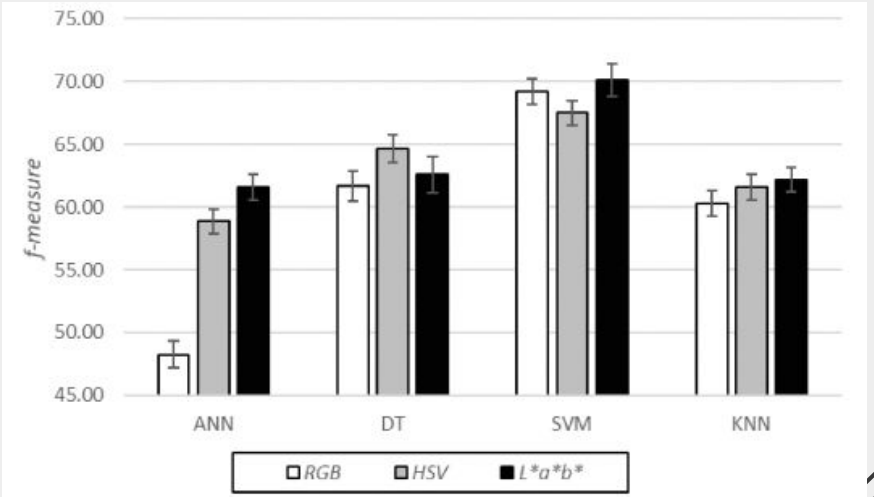
Ripeness states of Cape gooseberry



Preprocessing stages



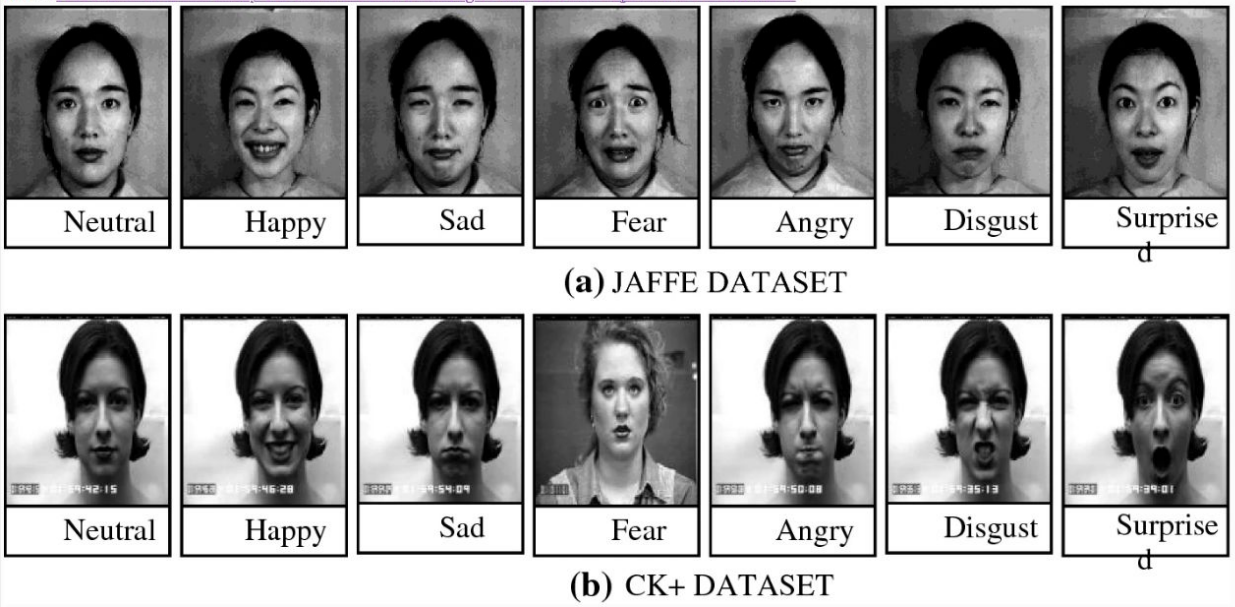
Results



Machine Learning Model for Face Expression Classification

Fig. 8

From: Multi-Featured and Fuzzy-Filtered Machine Learning Model for Face Expression Classification



My Method

PREPROCESSING / TRAINING

```
def fruits(data, data_type, photo_type):
    if photo_type == 'Gray':
        list_of_img = []
        list_of_labels = []
        # Change this to match your path
        path = "../Code/fruits-360/" + data_type + "/"
        # Looping through i which is the label and j is the specific photo.
        for i, j in enumerate(data):
            # Add the path plus the photo
            p = path + j
            # glob.glob() is recursively finding
            for x in glob.glob(os.path.join(p, "*.jpg")):
                # Read in the photo with a greyscale
                img = cv2.imread(x, cv2.IMREAD_GRAYSCALE)
                # Resizing the photo to 28x28 because thats how the paper did theirs
                img = cv2.resize(img, (dim, dim))
                # This kept it grey scale without it, the photo was not true greyscale
                img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
                # This is to reduce the noise in the dataset
                img = cv2.GaussianBlur(img, (5, 5), cv2.BORDER_DEFAULT)
                # Add the image to list
                list_of_img.append(img)
                # Add the label to list
                list_of_labels.append(i)

        list_of_img = np.array(list_of_img)
        list_of_labels = np.array(list_of_labels)

        return list_of_img, list_of_labels
    else:
        list_of_img = []
        list_of_labels = []
        # Change this to match your path
        path = "../Code/fruits-360/" + data_type + "/"
        # Looping through i which is the label and j is the specific photo.
        for i, j in enumerate(data):
            # Add the path plus the photo
            p = path + j
            # glob.glob() is recursively finding
            for x in glob.glob(os.path.join(p, "*.jpg")):
                # Read in the photo with a greyscale
                img = cv2.imread(x, cv2.IMREAD_GRAYSCALE)
                # Resizing the photo to 28x28 because thats how the paper did theirs
                img = cv2.resize(img, (dim, dim))
                # Turn to Binary greyscale
                img = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
                # This is to reduce the noise in the dataset
                img = cv2.GaussianBlur(img, (5, 5), cv2.BORDER_DEFAULT)
                 # This kept it grey scale without it, the photo was not true greyscale
                img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
                # Add the image to list
                list_of_img.append(img)
                # Add the label to list
                list_of_labels.append(i)

        list_of_img = np.array(list_of_img)
        list_of_labels = np.array(list_of_labels)
```

MANIPULATING MY DATASET

MNIST

```
# Training the data using train_test_split
X = digits_dataset.data
T= digits_dataset.target
X_train, X_test, t_train, t_test = train_test_split(X, T, test_size=0.25, shuffle=True)
```

Fruits 360

```
In [52]: # The fruit I want to work with - all picked for their similar dataset size
         fruit_data = ['Banana', 'Cocos', 'Mandarine', 'Pineapple', 'Raspberry', 'Dates', 'Apple Red Delicious', 'Cactus fruit', 'Clementi
```

```
In [53]: # TRaining the X_training and t_training on the training dataset
         X_training, t_training = fruits(fruit_data, 'Training', 'Gray')

         # X_test and t_test on the test dataset
         X_test, t_test = fruits(fruit_data, 'Test', "Gray")
```

```
In [55]: # Creating a standard scalar instance
         scaler = StandardScaler()

         X_training = scaler.fit_transform([i.flatten() for i in X_training])
         X_test = scaler.fit_transform([i.flatten() for i in X_test])
```

EXAMPLE OF ONE MODEL

In [56]:
```
# To start the timer
start_time = time.time()

# Creating the support vector machine instance
svm = SVC()

# Fitting the svm model
svm.fit(X_training, t_training)

# Predicting the model
predicted_svm = svm.predict(X_test)

# Training the model and printing out the scores
train_score_svm = svm.score(X_training, t_training)
test_score_svm = svm.score(X_test, t_test)
print("Train Accuracy: {}, Test Accuracy: {}".format(train_score_svm, test_score_svm))

# The Report to see exactly how the model is doing
print("\nClassification report for %s:\n%s\n" % (svm, metrics.classification_report(t_test, predicted_svm, target_names=fruit_dat

# To print out how long it took to run
print("Grayscale SVC took", time.time() - start_time, "to run")
```

```
Train Accuracy: 0.9991836734693877, Test Accuracy: 0.9451807228915663

Classification report for SVC():
                     precision    recall  f1-score   support

             Banana       1.00      0.83      0.91       166
              Cocos       0.99      0.92      0.95       166
          Mandarine       1.00      0.82      0.90       166
          Pineapple       1.00      0.98      0.99       166
          Raspberry       0.92      0.99      0.96       166
              Dates       0.91      1.00      0.95       166
Apple Red Delicious       1.00      0.92      0.96       166
       Cactus fruit       0.82      1.00      0.90       166
        Clementine        0.88      1.00      0.94       166
        Granadilla        1.00      1.00      1.00       166

           accuracy                           0.95      1660
          macro avg       0.95      0.95      0.95      1660
       weighted avg       0.95      0.95      0.95      1660


Grayscale SVC took 24.51749873161316 to run
```

My Results

MY RESULTS WITH THE MNIST DATASET

Table 2: Classification Report (F1 Score, Recall and Precision)

Name of Algorithms	F1 Score	Recall	Precision
Decision Tree Classifier	0.91	0.90	0.90
SVM	0.96	0.95	0.95
Random Forest Classifier	0.94	0.93	0.93
Naïve Bayes	0.87	0.87	0.86
KNN	0.86	0.85	0.84

Name of Algorithms	**F1 Score**	**Recall**	**Precision**	**Time-Taken (in seconds)**
Decision Tree Classifier	0.84	0.84	0.84	0.11
SVM	0.99	0.99	0.99	0.13
Random Forest Classifier	0.95	0.95	0.95	0.35
Naïve Bayes	0.84	0.84	0.87	0.11
KNN	0.99	0.99	0.99	0.98

FRUIT 360 GRAYSCALE

Table 2: Classification Report (F1 Score, Recall and Precision)

Name of Algorithms	F1 Score	Recall	Precision
Decision Tree Classifier	0.91	0.90	0.90
SVM	0.96	0.95	0.95
Random Forest Classifier	0.94	0.93	0.93
Naïve Bayes	0.87	0.87	0.86
KNN	0.86	0.85	0.84

Algorithms for Grayscale	F1 Score	Recall	Precision	Time-Taken (in seconds)
Decision Tree	0.71	0.71	0.75	3.52
SVM	0.95	0.95	0.95	24.51
Random Forest	0.84	0.84	0.87	0.55
Naïve Bayes	0.62	0.61	0.65	0.28
KNN	0.94	0.94	0.95	1.58

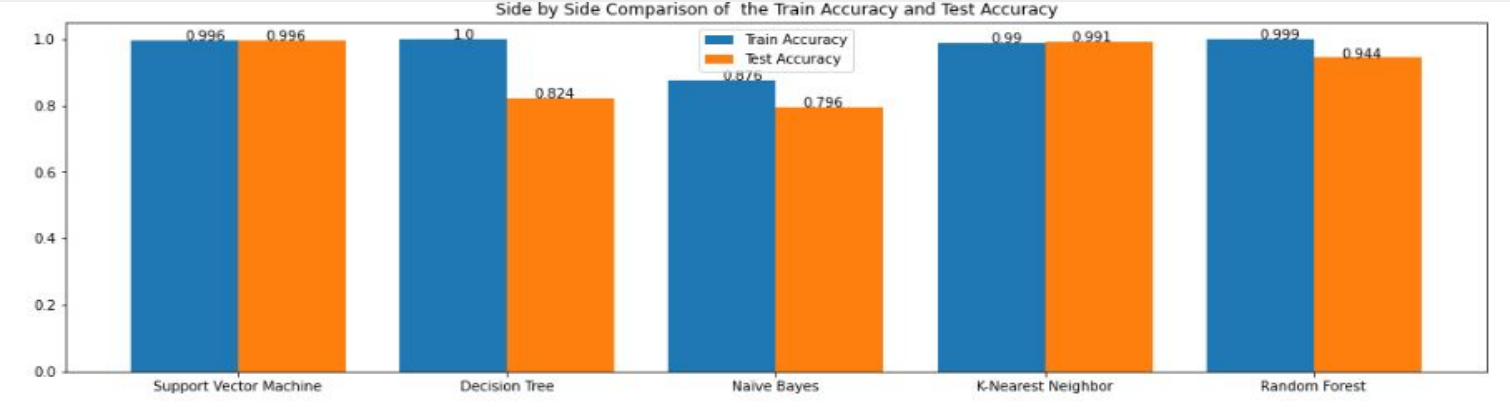
FRUITS 360 BINARY GRAYSCALE

Table 2: Classification Report (F1 Score, Recall and Precision)

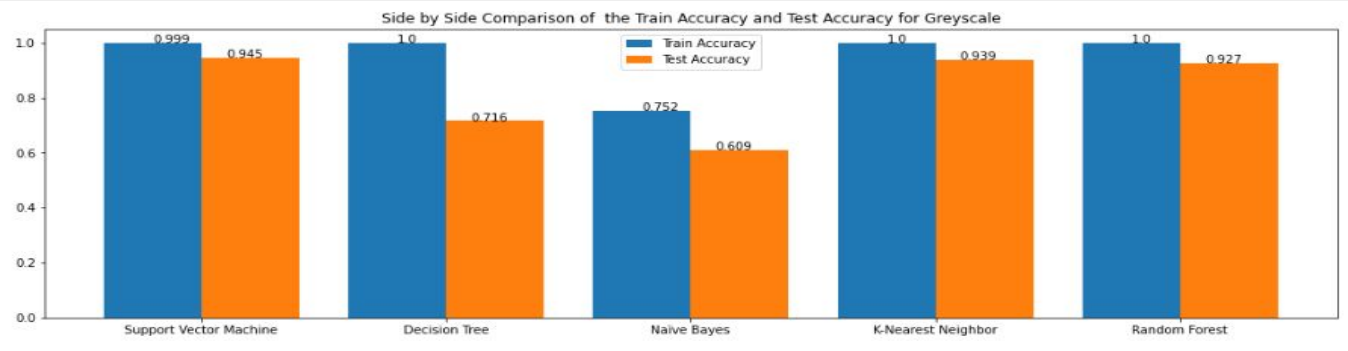
Name of Algorithms	F1 Score	Recall	Precision
Decision Tree Classifier	0.91	0.90	0.90
SVM	0.96	0.95	0.95
Random Forest Classifier	0.94	0.93	0.93
Naïve Bayes	0.87	0.87	0.86
KNN	0.86	0.85	0.84

Algorithms for Binary	F1 Score	Recall	Precision	Time-Taken (in seconds)
Decision Tree	0.68	0.68	0.71	4.51
SVM	0.92	0.92	0.92	33.90
Random Forest	0.83	0.83	0.85	0.67
Naïve Bayes	0.58	0.58	0.63	0.28
KNN	0.92	0.92	0.93	1.60

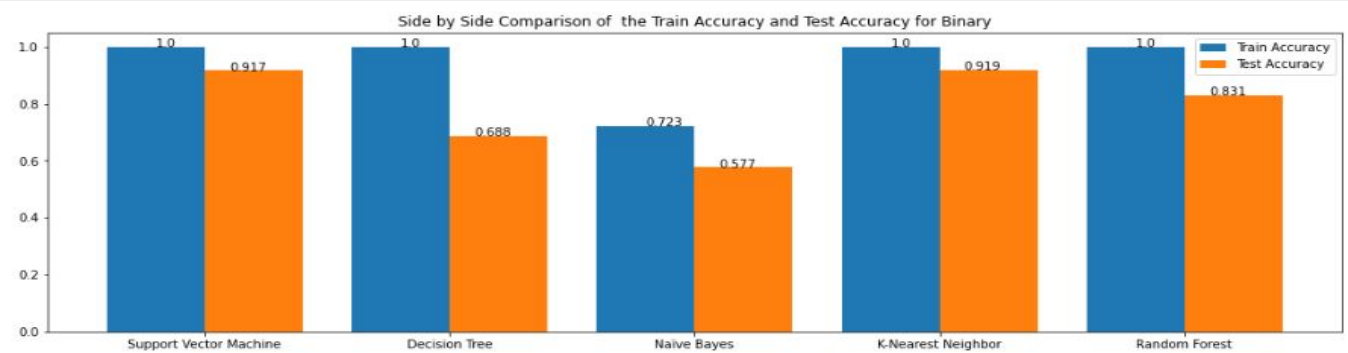
MNIST RESULTS



GRAYSCALE FRUITS



BINARY FRUITS



Observations and Future Fixes

- Lower my models processing times
- Continue fine-tuning the preprocessing of my dataset
- Experiment with more fruits and various training and testing dataset sizes

Conclusion

I had a lot of fun and to be honest some frustration with this project. I learned a lot about how to pull in large datasets, manipulate the dataset to print out its photos, and it ensure it is in the correct formatting to be passed into the models. This is something that had exceeded my expectations in difficulty and how advanced our technology is currently, I thought this would be a simpler task. However, for my skill set for this project I was super proud of the outcome. It was interesting to learn how to use cv2 library to manipulate the color values and use Gaussian Blur to remove noise from the photos and to figure out ways to improve the models efficiency by tweaking with the pre-processing method of the dataset.

THANK YOU!

CITATIONS

Gope, Birjit, et al. "Handwritten Digits Identification using Mnist Database Via Machine Learning Models." IOP Conference Series.Materials Science and Engineering, vol. 1022, no. 1, 2021. ProQuest,doi:http://dx.doi.org/10.1088/1757-899X/1022/1/012108.

Juneja, Kapil, and Chhavi Rana. “Multi-Featured and Fuzzy-Filtered Machine Learning Model for Face Expression Classification.” *Wireless Personal Communications*, vol. 115, no. 2, Springer US, 2020, pp. 1227–56, https://doi.org/10.1007/s11277-020-07620-8.

W. Castro, J. Oblitas, M. De-La-Torre, C. Cotrina, K. Bazán and H. Avila-George, "Classification of Cape Gooseberry Fruit According to its Level of Ripeness Using Machine Learning Techniques and Different Color Spaces," in IEEE Access, vol. 7, pp. 27389-27400, 2019, doi: 10.1109/ACCESS.2019.2898223.

Dataset link: https://www.kaggle.com/moltean/fruits and http://yann.lecun.com/exdb/mnist/