TEAM 1

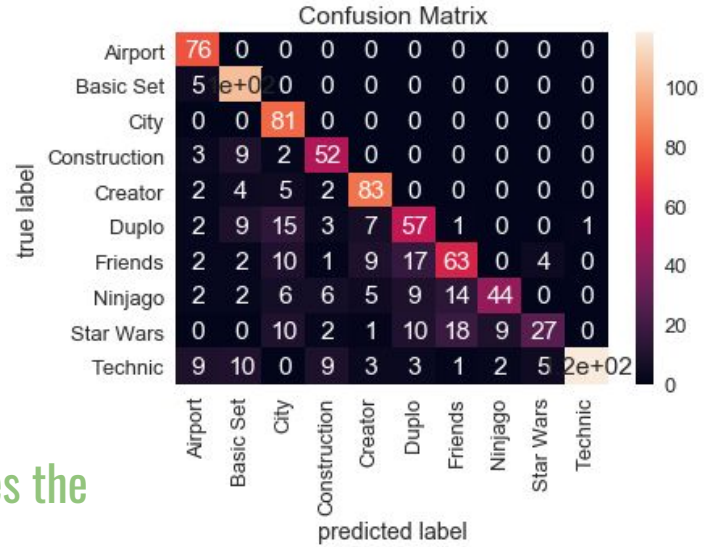
Chi, Drystan, Erica, James M., James R., Marie

kNN

1) **Parameters Tuned:** We Played with n to see what number gave a better score, n=2 gave the highest performance. After trying different colours for prediction parameters, we settled on using a few of the most popular numbers and a few of the least used colours.

2) Performance

- Accuracy = 0.7374476987447699
- Precision = 0.75513999200747683
- Recall = 0.7374476987447699
- F1 = 0.726961113572522O2



3) Interpretation:

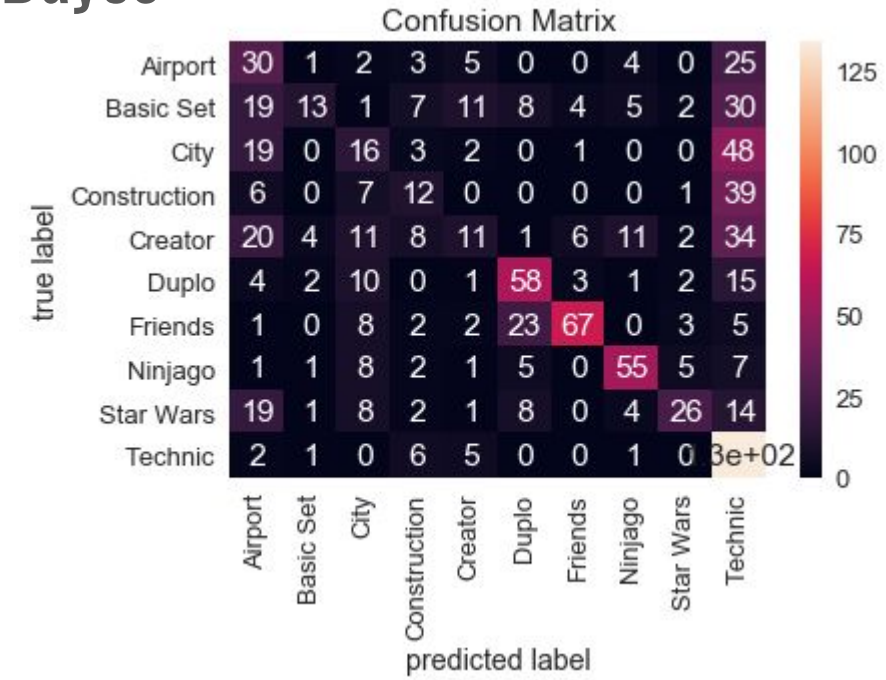
When there are too many neighbors evaluated, it reduces the performance. We found the higher n was, the lower our performance scores got. This was true even we changed what colours were used.

Naïve Bayes

Parameters: All Colours

Performance

- Accuracy = 0.44142259414225943
- Precision = 0.47576859278626743
- Recall = 0.44142259414225943
- F1 = 0.41390912706962701



Interpretation: Using varying combinations of Year, Num Parts, and colour still resulted in performance values (< 50%)

Decision Tree

1) Parameters

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
vect = TfidfVectorizer()
```

```
lego_dtree_frame['Set Name Vect'] = list(vect.fit_transform(lego_dtree_frame['Set Name']).toarray())
lego_dtree_frame.sample(1)
```

```
predictors = lego_dtree_frame['Set Name Vect'].tolist()
```

```
predictee = lego_dtree_frame['Theme']
```

2) Performance

Accuracy = 0.7018828451882845
Precision = 0.7580672353860669
Recall = 0.7018828451882845
F1 = 0.7137163328832542

Decision Tree

3) Interpretation

```
lego_dtree_frame['Set Name Vect'].sample(5)
```

```
1674    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
3347    [0.405040654764453, 0.0, 0.0, 0.0, 0.0, 0.0, 0...
6327    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
4103    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
7698    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ...
Name: Set Name Vect, dtype: object
```

TF-IDF -> deals with words
Term Frequency:
    Raw Count
Inverse Document Frequency:
    Value of information

TF-IDF = TF * IDF

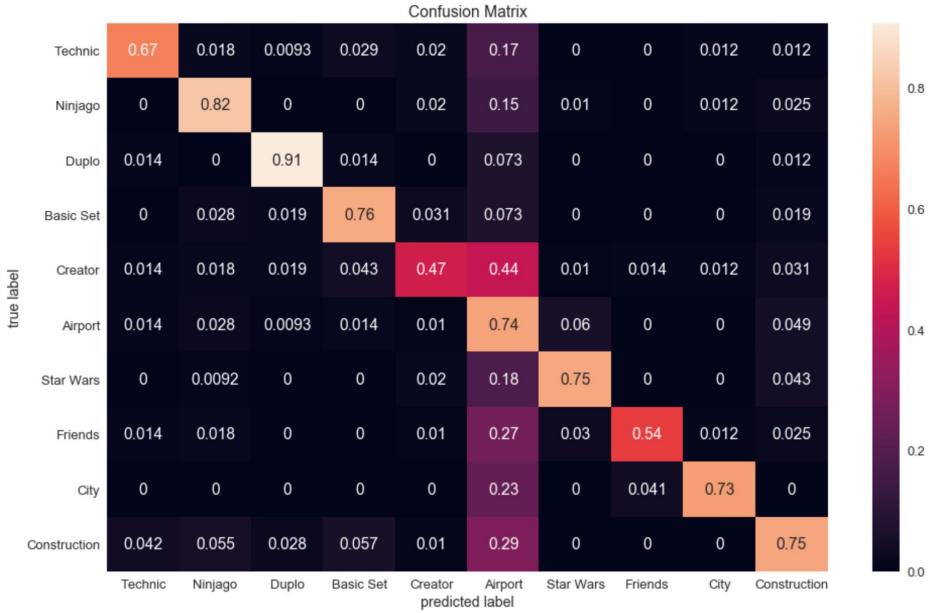
Predicts well for the themes of Duplo(>90%), Ninjago(>80%)
Has fairly strong metric scores which suggests it is an acceptable classifier.

Random Forest

Parameters: All colours

Performance

- Accuracy = 0.7039748953974896
- Precision = 0.7097356446898687
- Recall = 0.7039748953974896
- F1 = 0.6997997309302797



Interpretation

Duplo and Ninjago are well predicted

Neural Network

1) Parameters

```
# I applied the Neual Net across all the columns
size = 131
# I altered the layer sizes and max_iter. However this seeamed the maxium
# my Laptop would handle
mlp = MLPClassifier(hidden_layer_sizes=(2*size,4*size, 16*size),max_iter=500)
```

Neural Network

2) Performance

Using a 40/60 split I had the following classification report

	precision	recall	f1-score	support
Airport	0.57	0.64	0.60	61
Basic Set	0.83	0.71	0.76	107
City	0.54	0.44	0.49	115
Construction	0.72	0.65	0.69	75
Creator	0.44	0.48	0.46	97
Duplo	0.69	0.85	0.76	87
Friends	0.83	0.88	0.85	106
Ninjago	0.90	0.84	0.87	77
Star Wars	0.67	0.55	0.60	80
Technic	0.76	0.85	0.81	148
avg / total	0.70	0.70	0.69	953

Neural Network

Neural Network

3) Interpretation

I had good performance on the following types:
Friends,Duplo,Ninjago and Technic with a precision >= 0.84

TEAM 2

Andrew, Brenda, Hannah, Jason, Kent, Michael

1) kNN Classifier

N = 5

Database:

Normalized LEGO

Parameters:

- Light Gray
- Pearl Gold
- Black
- Bright Green
- Yellow
- White
- Medium Azure
- [No Color]
- Red
- Blue
- Light Bluish Gray
- Dark Bluish Gray
- Green

Scores

Accuracy: 0.60878
Precision: 0.60269
Recall: 0.60878

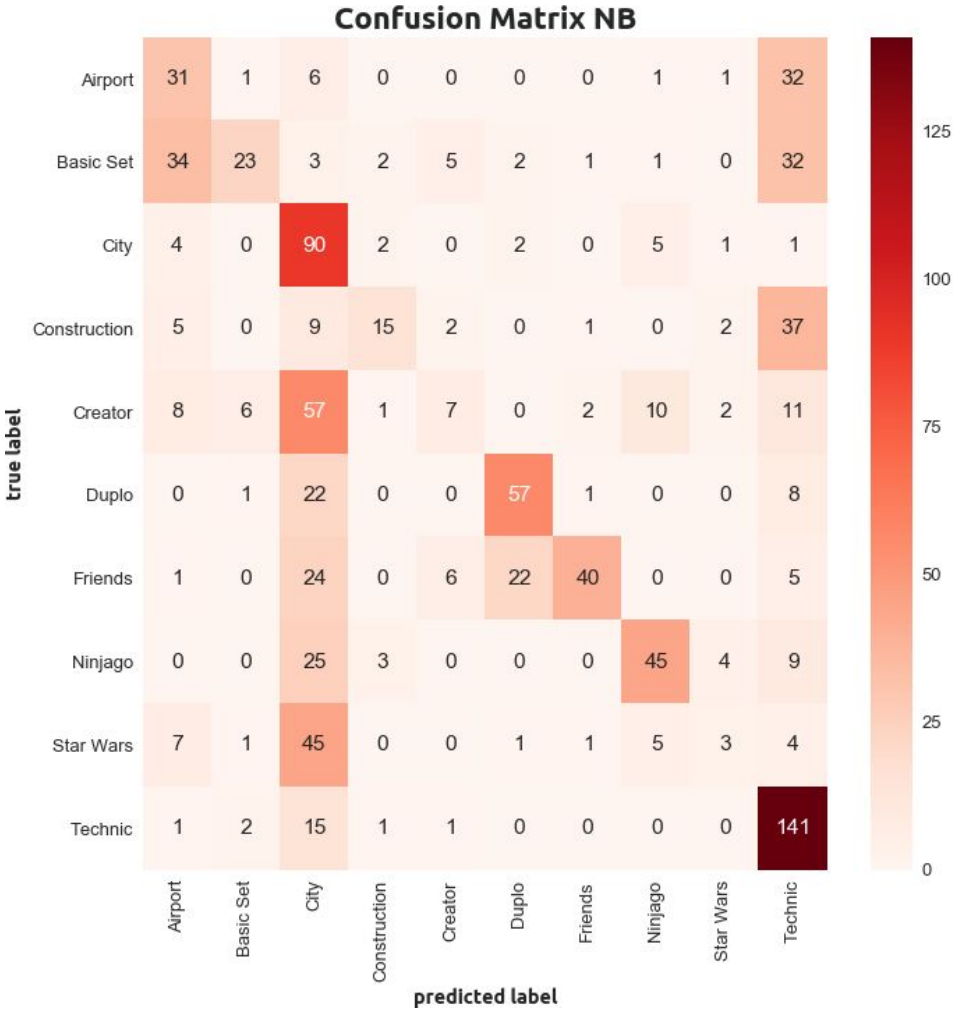
F1: 0.60039

2) Naïve Bayes

Scores

Accuracy: 0.58159
Precision: 0.58540
Recall: 0.58159

F1: 0.57758



Database:

LEGO_frame

Parameters:

- Year
- Red
- Blue
- Black
- Yellow

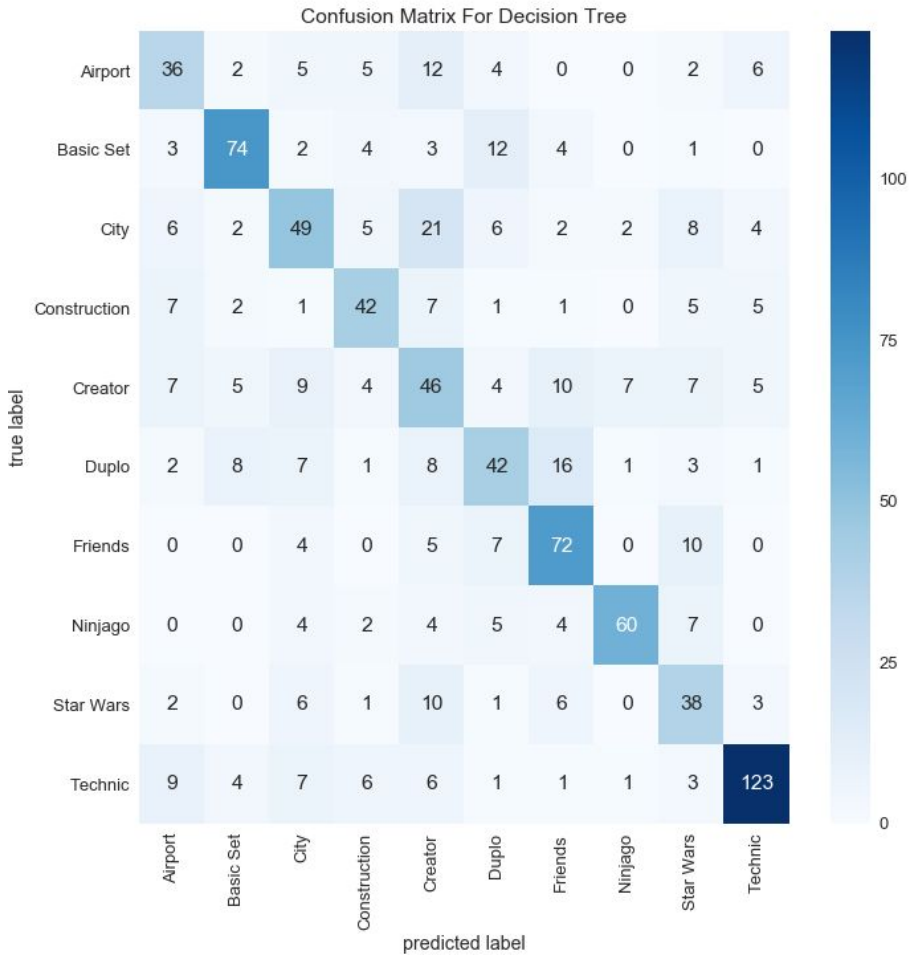
3) Decision Tree

Scores

Accuracy: 0.611924
Precision: 0.624278
Recall: 0.611924

F1: 0.614624

Observations:
Basic Sets and Friends excelled > 71
Technic was very high
NinjaGo was average
Airport and Star Wars were the worst < 39



Database:

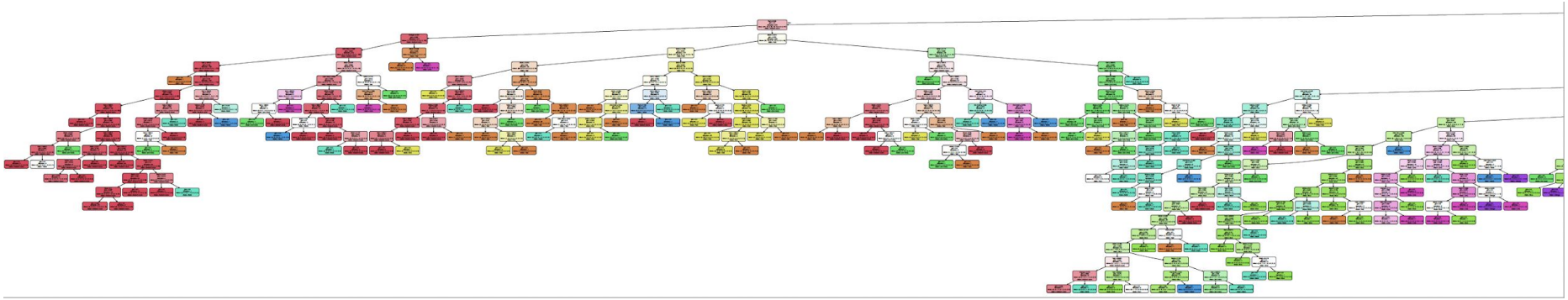
LEGO_frame

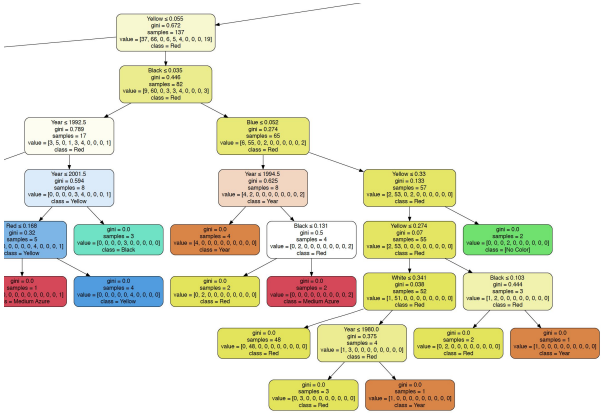
Parameters:

- Removed the off shades like trans-yellow, left the darks and specials like azure
- No color
- Year

"Traitor!"

Decision Tree Output and a crop out of the yellow branch





Star Wars

Why a massive score difference?

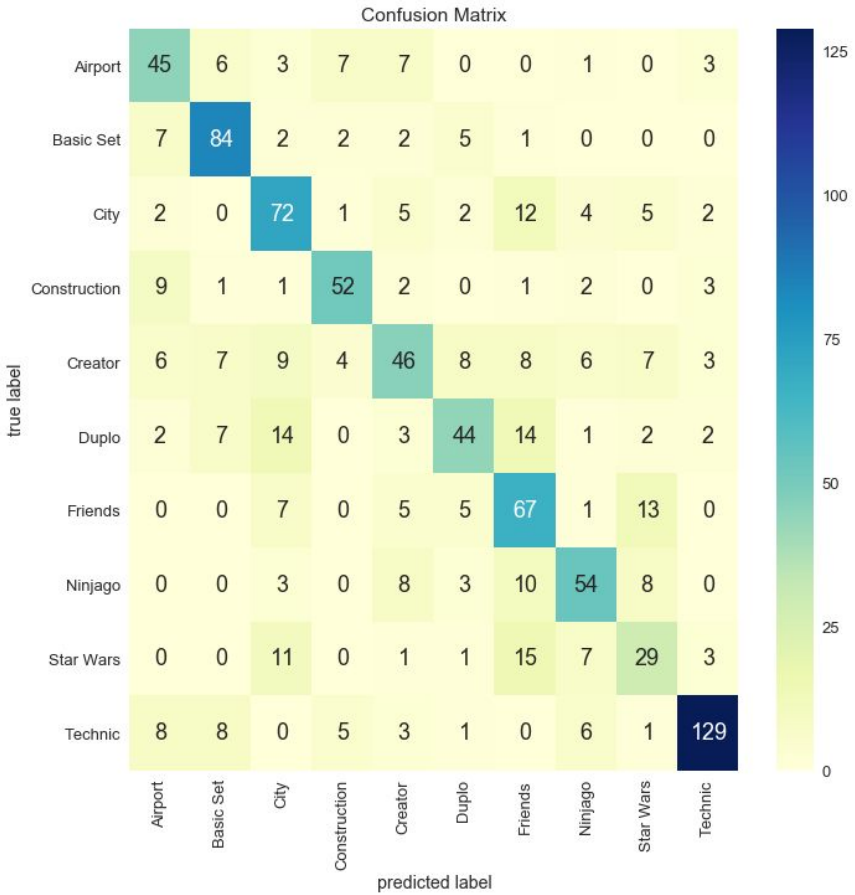
Technic

4) Random Forest Classifier

Scores

Accuracy: 0.70711
Precision: 0.71306
Recall: 0.70711

F1: 0.70565



Estimators = 100

Database:

Normalized LEGO

Parameters:

- [No Color]
- Black
- White
- Red
- Blue
- Yellow
- Green
- Year
- Num Parts

5) Neural Network Classifier

Database:

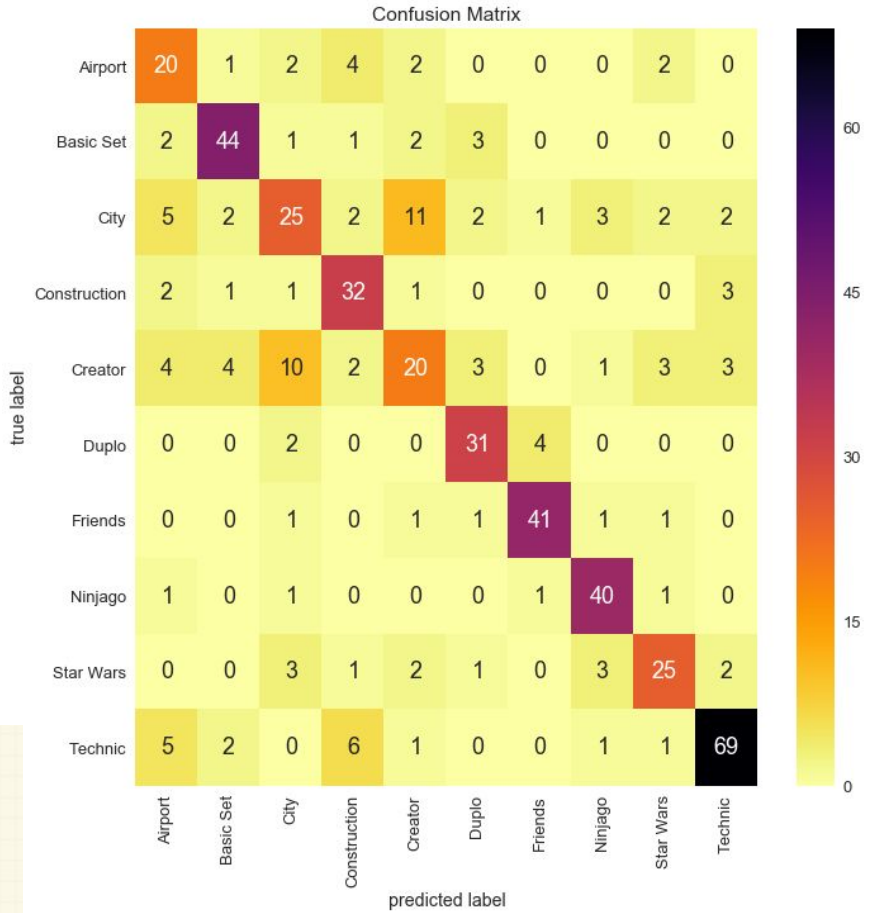
Normalized LEGO

Parameters:

- Red
- Yellow
- Green
- Blue
- White
- Black
- ....
- Every color except Very Light Orange
- Did not include Year or Num Parts

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.



Scores

Accuracy: 0.71757
Precision: 0.72172
Recall: 0.71757

F1: 0.71834

TEAM 3

Mason, Philmo, Shane, Shivani, Tyler

1. kNN:

Which parameters did I tune?

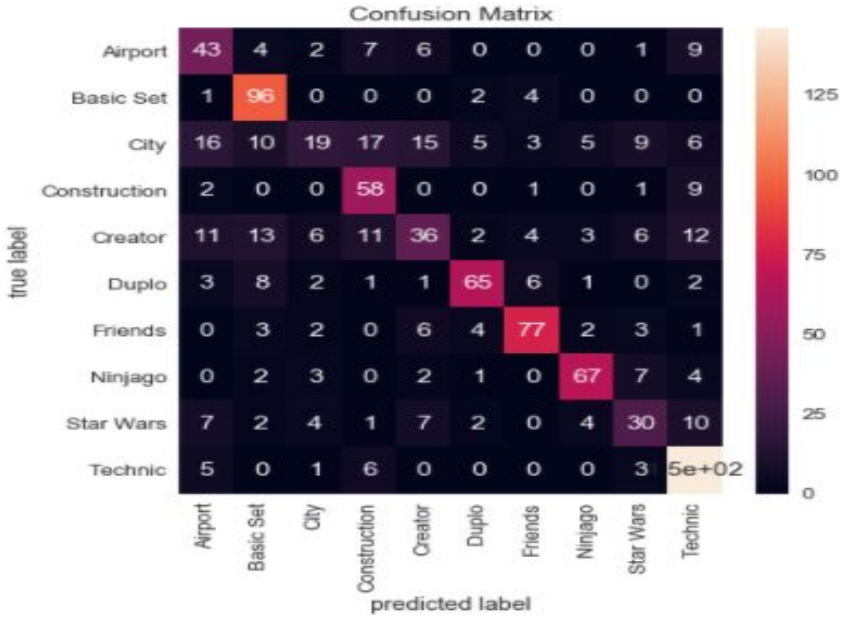
- Color columns with norm. values >= 0.1
- Num. of neighbors = 6
- *Average* metric = “micro”

Performance:

- Accuracy: 0.666317991631799 17
- Precision: 0.661075516162373 59
- Recall: 0.666317991631799 17
- **F1: 0.666317991631799 17**

Interpretation of Performance:

- Themes which seem to fare better are those that have more entries in the filtered out datasets
- Determination of neighbors value (n) -
  - Too small: overfitting
  - Too big: too little variance to be meaningful



2. Naïve Bayes

Performance

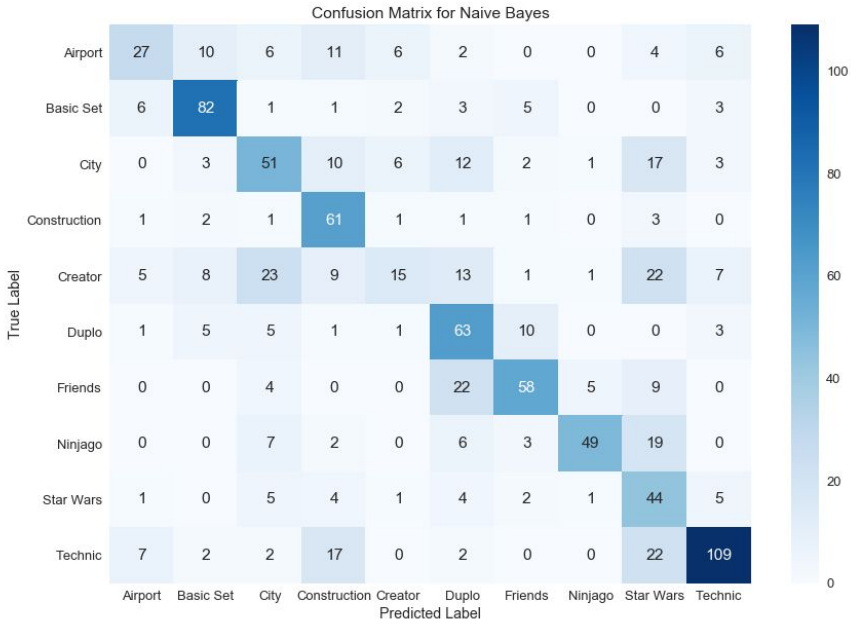
Accuracy:	0.5847280334728
Precision:	0.61730378031112
Recall:	0.5847280334728
F1:	**0.57704779053976**

Feature Selection

- Colors: Blue, Red, Yellow, White ,Black, Gray, Magenta, Gold
- Year and Number of Parts (normalized)

Parameters Tuned

- Colors grouped by their common name.
Eg. 'Chrome Pink' and 'Bright Pink' were grouped together under 'Pink'



Interpretation of Results

- ‘Creator’ being wrongly classified as ‘City’ was the biggest misinterpretation
- ‘Star Wars’ was most wrongly predicted for a given set
- These themes share similar color palettes and color ratios

3. Decision Tree Classifier

Performance

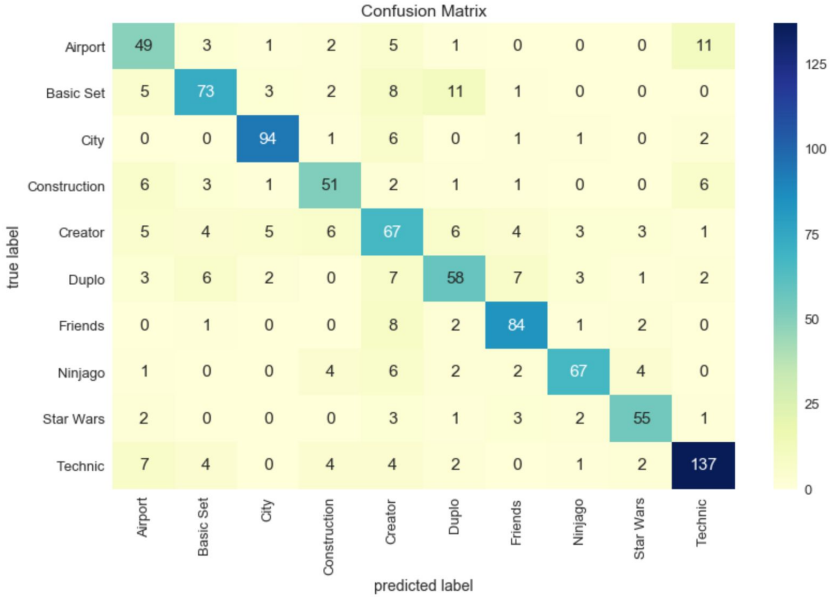
Accuracy:	0.768828451883
Precision:	0.772210911451
Recall:	0.768828451883
F1:	**0.769789243434**

Feature Selection

- Numerical Encoding of ‘Set Name’
- All colour names, ‘Year’, and ‘Num Parts’

Parameters Tuned

- Default hyper-parameters
- Learned model in 10 000 iteration loop, saved model with highest F1



Interpretation of Results

- High misclassification of ‘Creator’ (49 FP/ 42%)
- Low misclassification of ‘Technic’ (23 FP/ 14%) and ‘City’ (12FP/ 10.5%)
- Decision tree too deep to be highly interpretable

4. Neural Network Classifier

Performance

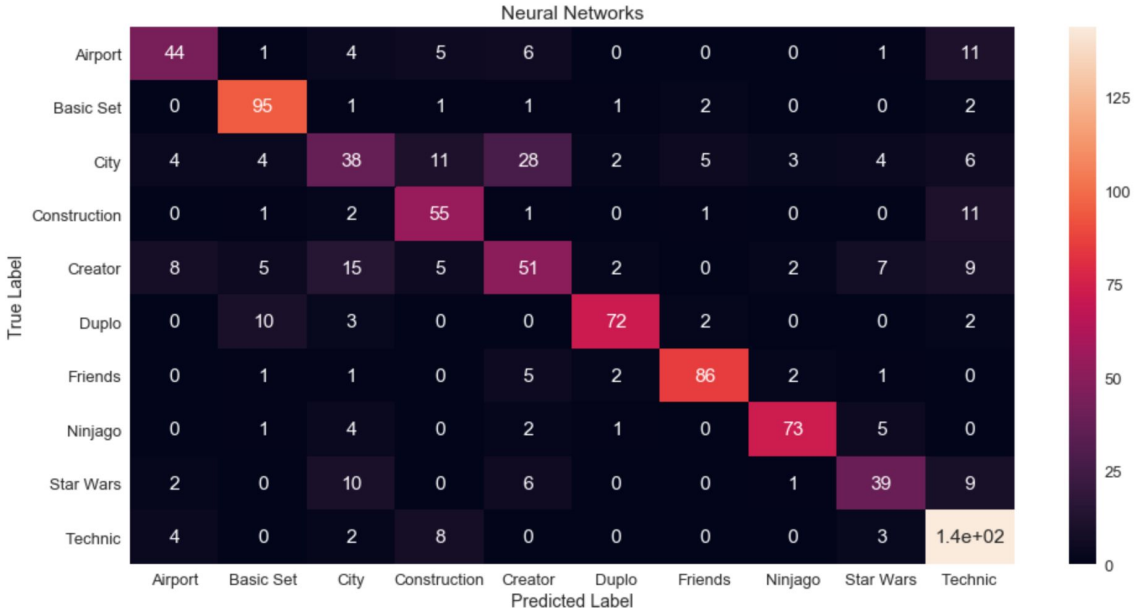
Accuracy:	0.72907949790794979
Precision:	0.72314819416636411
Recall:	0.72907949790794979
F1:	**0.72249486715630373**

Feature Selection

- Colour names only

Parameters Tuned

- max_iter=1000
- default 200 node hidden layer



Interpretation of Results

- Very poor at classifying 'Star Wars' theme (35% FPs)
- overall quite average performance. Same problems as other classifiers

5. Random Forest

Performance

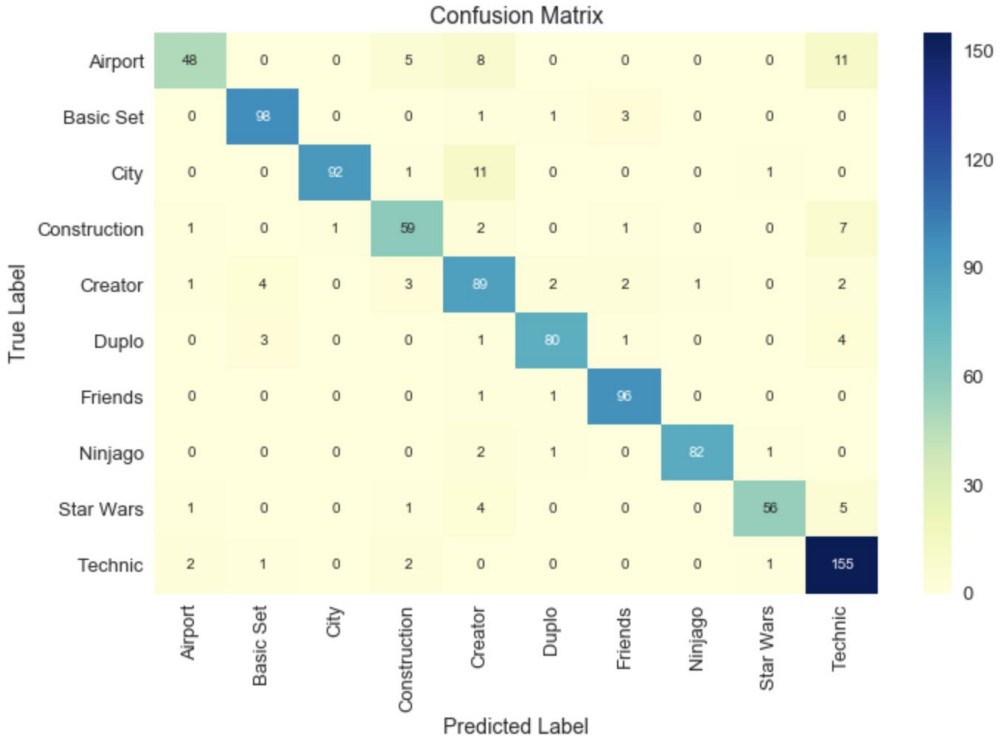
Accuracy:	0.89435146
Precision:	0.89997178
Recall:	0.89435146
F1:	**0.89394077**

Feature Selection

- Numerical Encoding of ‘Set Name’
- All colour names, ‘Year’, and ‘Num Parts’

Parameters Tuned

- n_estimators, max_features, max_depth, bootstrap, random_state



Interpretation of Results

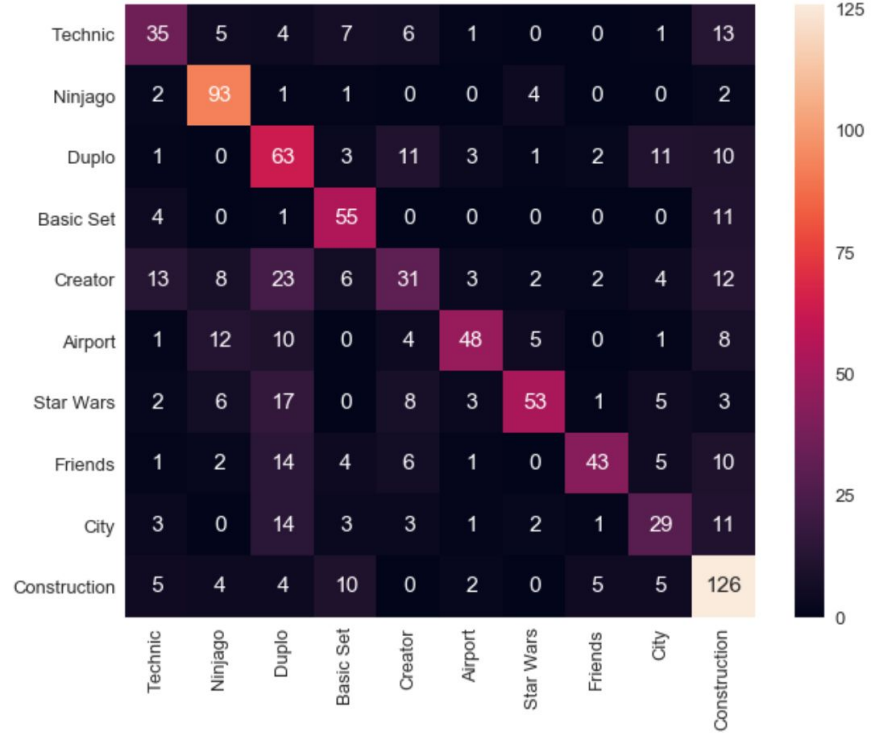
- High misclassification of ‘Creator’ (30 FP / 25%)
- High misclassification of ‘Technic’ (29 FP / 16%)
- Lowest misclassification of ‘City’ (1 FP / 0.012%)

TEAM 4

Alex, Autism, Debris, Edward Elric, Phteven

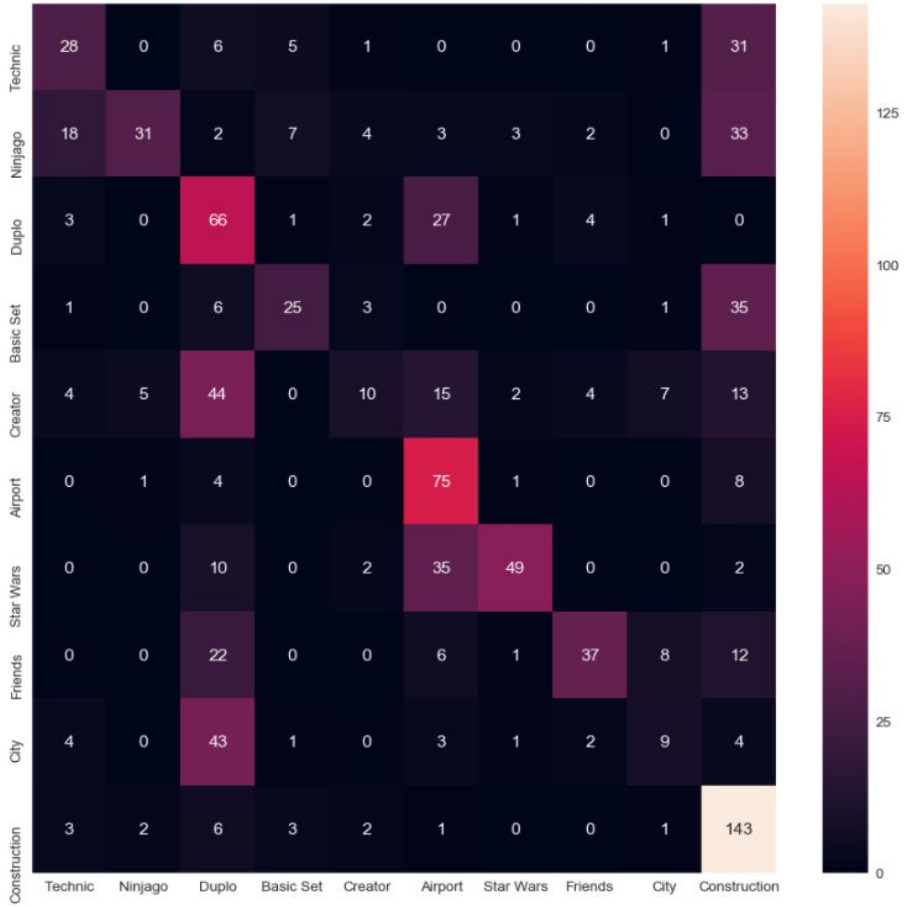
k, nearst neighbe

```
all_colors = lff[lff.columns.drop(['Set Num', 'Set Name', 'Theme', 'Year', 'Num Parts', 'Unknown'])]

# https://stackoverflow.com/questions/21164910/delete-column-in-pandas-if-it-is-all-zeros
# Remove columns with less than 5 values
colors = all_colors.loc[:, (all_colors > 5).any(axis=0)]

lff_train, lff_test, pos_train, pos_test = train_test_split(colors, lff['Theme'], test_size=0.4, random_state=0)

lknn_class = KNeighborsClassifier(n_neighbors=6)
lknn_class.fit(lff_train, pos_train)
```



Accuracy: 0.602510460251
Precision: 0.618138940793
Recall: 0.602510460251
F1: 0.595737180894

2. Naive Bayes




```
Precision score:    0.549506936268
Accuracy score:     0.494769874477
Recall score:       0.494769874477
F1 score:           0.46572048421
```

Counted the amount of times a color appeared in a lego set and picked the top 30

Also added Years and Num Parts

```
new_columns_nn.append("Black")
new_columns_nn.append("White")
new_columns_nn.append("Red")
new_columns_nn.append("Yellow")
new_columns_nn.append("Blue")
new_columns_nn.append("Light Bluish Gray")
new_columns_nn.append("Dark Bluish Gray")
new_columns_nn.append("Trans-Clear")
new_columns_nn.append("Green")
new_columns_nn.append("Reddish Brown")
new_columns_nn.append("Trans-Red")
new_columns_nn.append("Orange")
new_columns_nn.append("Light Gray")
new_columns_nn.append("Lime")
new_columns_nn.append("Trans-Yellow")
new_columns_nn.append("Flat Silver")
new_columns_nn.append("Trans-Light Blue")
new_columns_nn.append("Pearl Gold")
new_columns_nn.append("Trans-Black")
new_columns_nn.append("Bright Green")
new_columns_nn.append("Trans-Orange")
new_columns_nn.append("[No Color]")
new_columns_nn.append("Dark Tan")
new_columns_nn.append("Bright Light Orange")
new_columns_nn.append("Medium Blue")
new_columns_nn.append("Medium Azure")
new_columns_nn.append("Dark Orange")
new_columns_nn.append("Dark Pink")
new_columns_nn.append("Trans-Dark Blue")
new_columns_nn.append("Trans-Green")
new_columns_nn.append("Num Parts")
new_columns_nn.append("Year")
```

3. Decision Trees

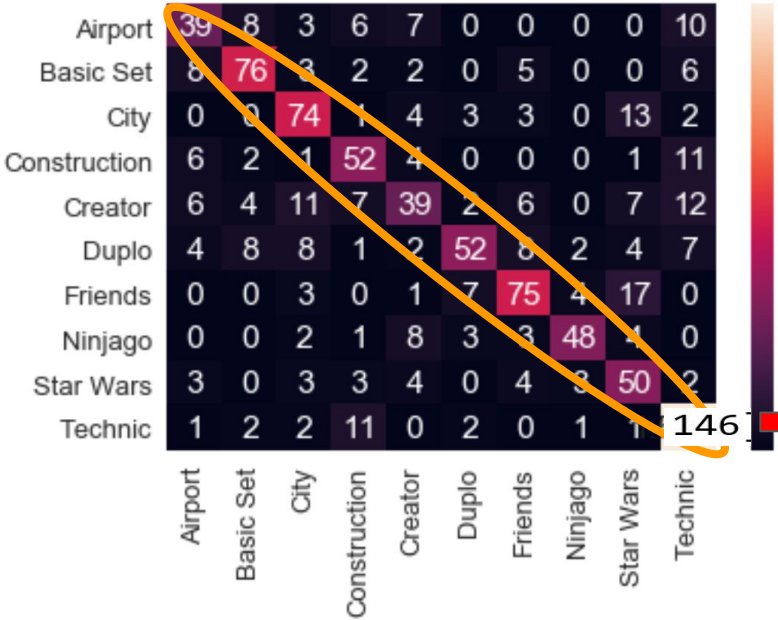
some_col consists of Year, Num Parts, colors

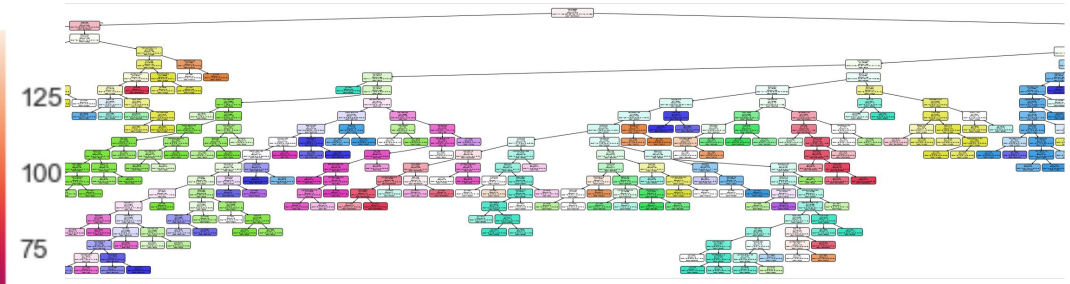
1

```
ynp_train, ynp_test, theme_train, theme_test = sk.model_selection.train_test_split(
    lego_frame[some_col],lego_frame['Theme'], test_size=0.4)

clf_gini = DecisionTreeClassifier(criterion = "gini",
                                  max_depth=20, min_samples_leaf=5)
clf_gini.fit(ynp_train, theme_train)
```

2

```
Accuracy: 0.6809623430962343
Precision: 0.6840658297671064
Recall Score: 0.6809623430962343
F1 Score: 0.6771052438109705
```





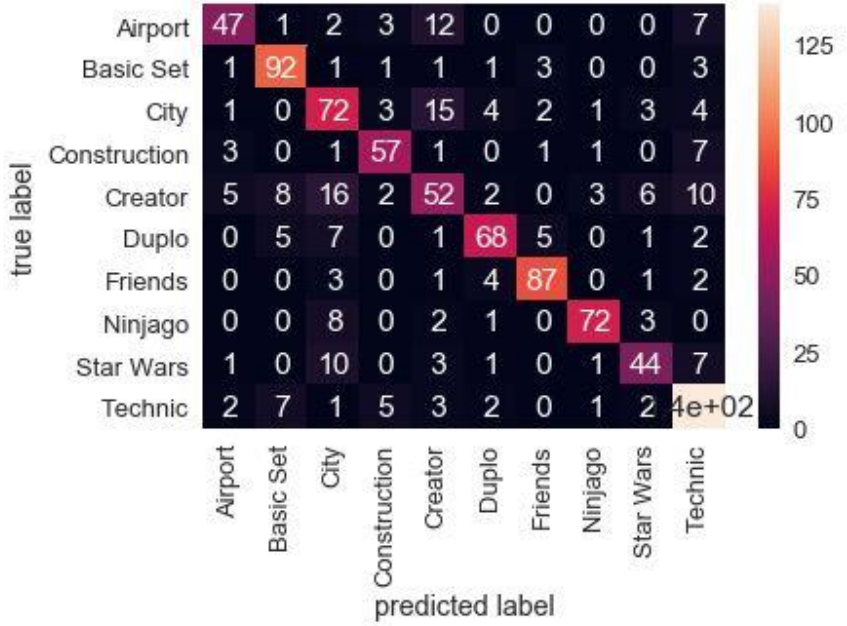
3

Best Themes: Technic -True Positives (88%) (146/166)
Ninjago - Precision (83%) (48/58)
Worst Themes: Airport - True Positives (53%) (39/73)
Star Wars - precision (50/97) - 52% (when predicting "Star Wars", classifier is correct 52% of the time)

4. Random Forest

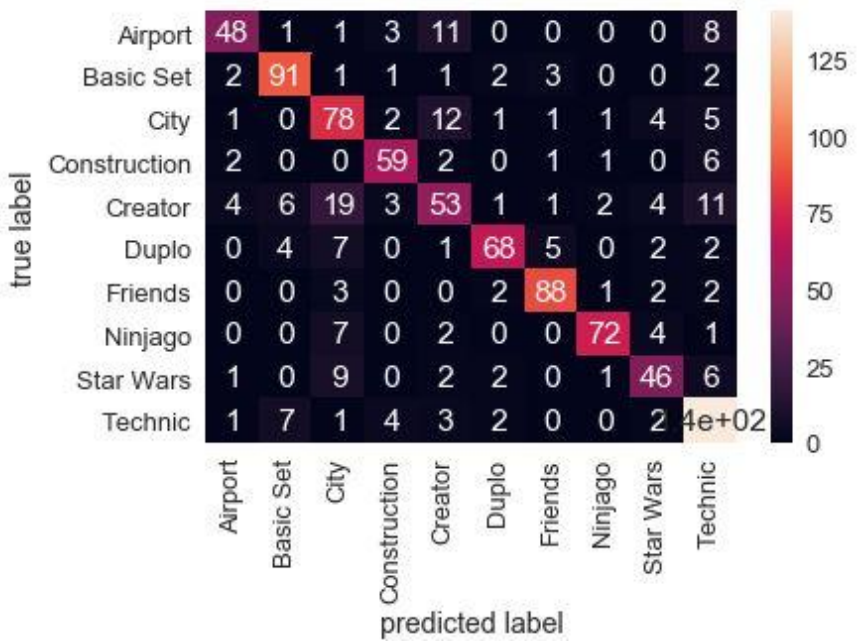
```
ssifier(n_estimators=100, random_state=0)
theme_train)
```

```
Accuracy: 0.7625523012552301
Precision: 0.7636280266777246
Recall: 0.7625523012552301
F1: 0.7610308319199467
```




```
ier(n_estimators=500, random_state=0)
e_train)
```

```
Accuracy: 0.7782426778242678
Precision: 0.7810885924211509
Recall: 0.7782426778242678
F1: 0.7767778658043261
```

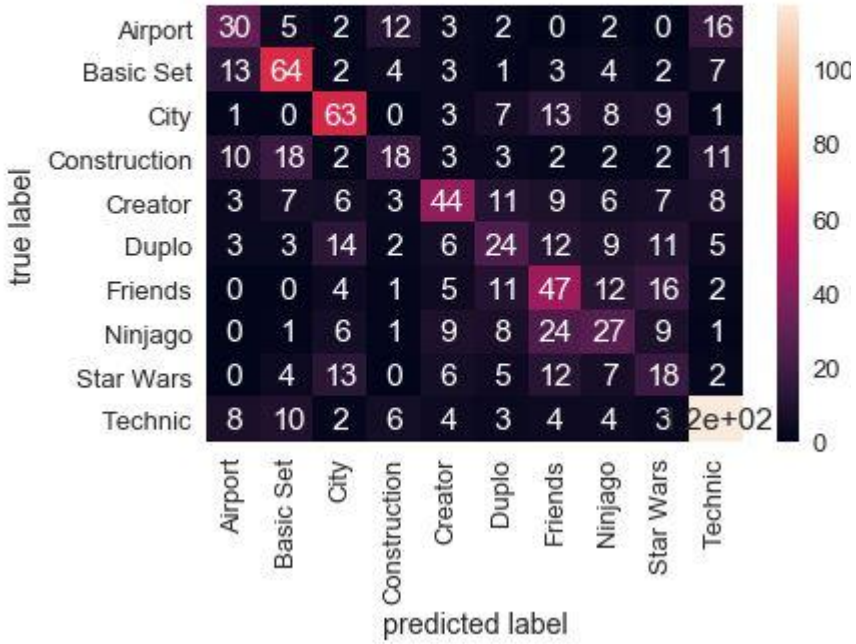


4. Random Forest

Using color performs better

```
parts_train, parts_test, theme_train, theme_test = sk.model_selection.train_test_
    lego_frame[['Num Parts', 'Year']],lego_frame['Theme'], test_size=0.4, random_
```

```
Accuracy: 0.47280334728033474
Precision: 0.46989476825706494
Recall: 0.47280334728033474
F1: 0.4685872017099972
```



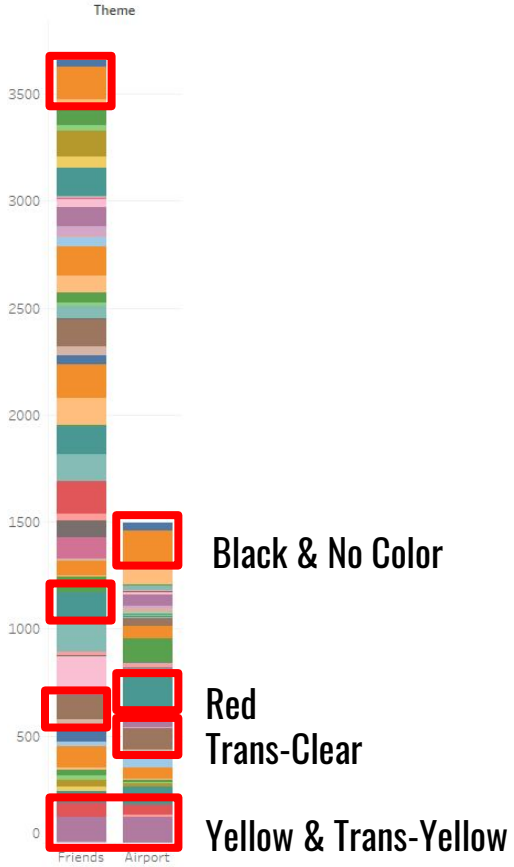
5. Neural Network

- Used all colour columns except White

```
MLPClassifier(
    hidden_layer_sizes=
        (70, 15, 20, 15)
    )
```

```
Precision score:        0.671966477808
Accuracy score:         0.678870292887
Recall score:           0.678870292887
F1 score:               0.665237527676
```

5. Neural Network: Why Airport and Friends?



Lots of similar colour counts?

TEAM 5

Bryson, Dianna, Flavia, Irene, Ray

K-Nearest Neighbour

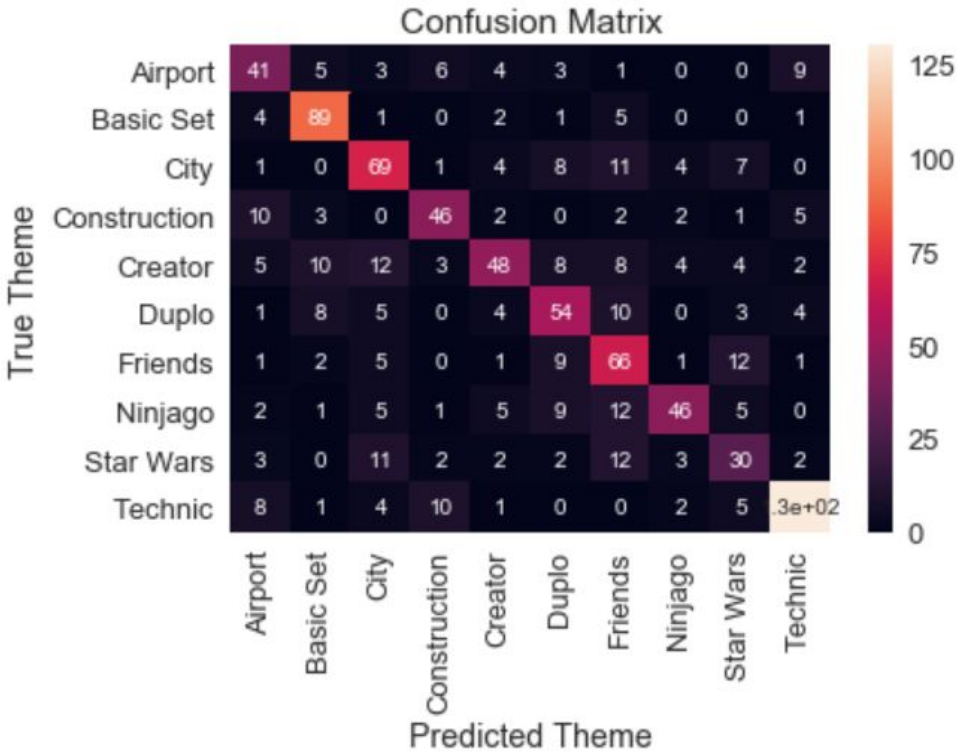
```
Accuracy: 0.647489539749
Precision: 0.655193461967
Recall: 0.647489539749
F1 Score: 0.646204297285
```

- 'Basic Set' has a consistent, wide variety of colours unlike most other sets which focus more on certain theme colors

-'Technic' is unique as it is a newer theme with a lot of small pieces, often of which a significant portion are yellow

-'Star Wars' often consists of a lot of gray which makes it hard to set it apart from the other sets with a common large portion of gray.

-'City' and 'Friends' likely often share similar amounts of the some same popular colours with other sets, making it harder to distinguish between 'City', 'Friends' and the other less accurately predicted themes.



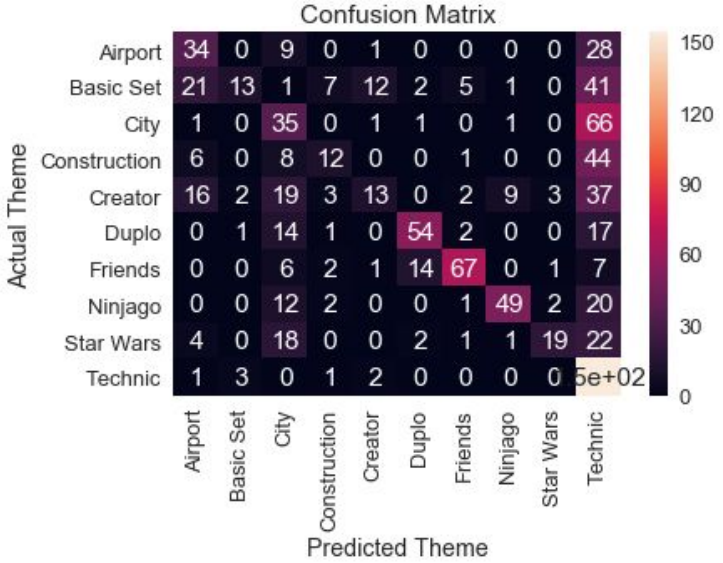

```
lego_train, lego_test, theme_train, theme_test = sk.model_selection.train_test_split(
    lego_frame[['Year','Num Parts','[No Color]','Black','Brown','Red','White','Yellow','Green','Blue', 'Dark Bluish Gray',
                'Dark Gray', 'Light Bluish Gray', 'Light Gray', 'Pearl Gold']],lego_frame['Theme'], test_size=0.4, random_state=0)
n=4
```

Naive Bayes

Accuracy rating increases depending on the number of values added.

Technic ruins the ratings of Naive Bayes due to most piece-colors primarily being common across all sets and themes.

Accuracy:	0.470711297071
Precision:	0.555862269046
Recall:	0.470711297071
F1 Score:	0.444458832026




```
## Tests/Trains with the specified values. (Top 3 colors of each set excluding top 10 overall colors.)
nbList = ['Year','Trans-Light Blue', 'Trans-Black', 'Trans-Red', 'Orange', 'Lime', 'Trans-Clear',
          'Trans-Orange', 'Reddish Brown', 'Dark Green','Bright Green', 'Medium Azure', 'Magenta',
          'Pearl Gold', 'Dark Tan', 'Flat Silver', 'Dark Red', 'Dark Gray']
```

Decision Tree Classifier

```
accuracy = skm.accuracy_score(y_true=theme_test,
                              y_pred=lego_predictions_test)
accuracy
```

```
0.44665271966527198
```

```
precision =  skm.precision_score(y_true=theme_test,
                                 y_pred=lego_predictions_test,
                                 average='weighted')
precision
```

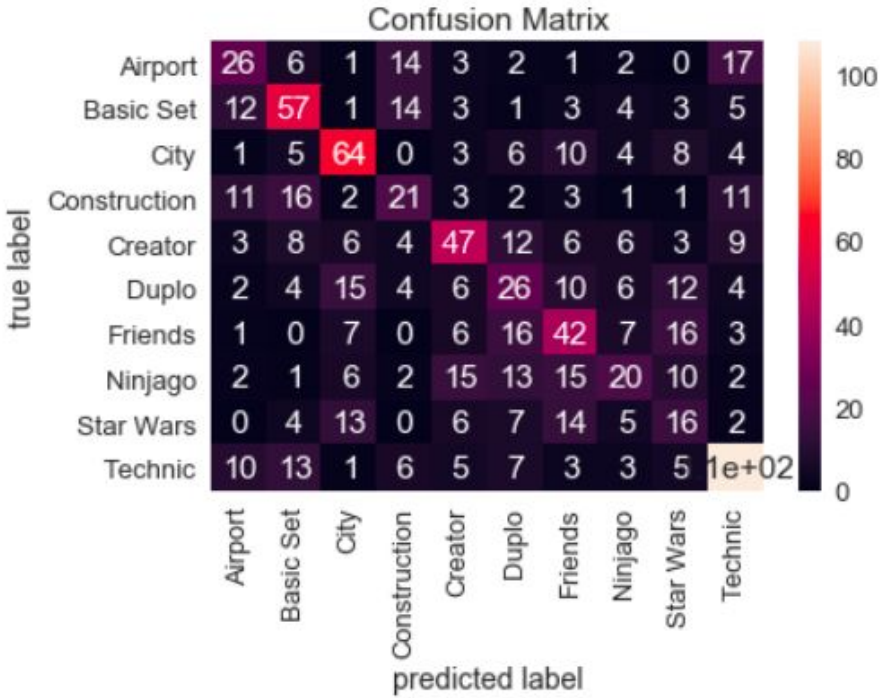
```
0.44292217685984631
```

```
recall = skm.recall_score(y_true=theme_test,
                          y_pred=lego_predictions_test,
                          average='weighted')
recall
```

```
0.44665271966527198
```

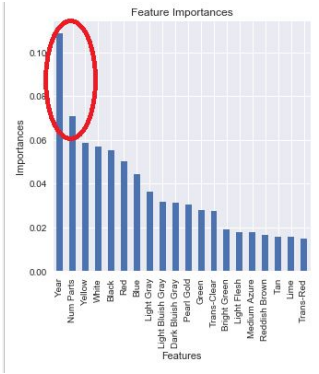
```
F1 = skm.f1_score(y_true=theme_test,
                  y_pred=lego_predictions_test,
                  average='weighted')
F1
```

```
0.44322781378223619
```



Random Forest Classifier: What features have the most effect?

Feature Selection



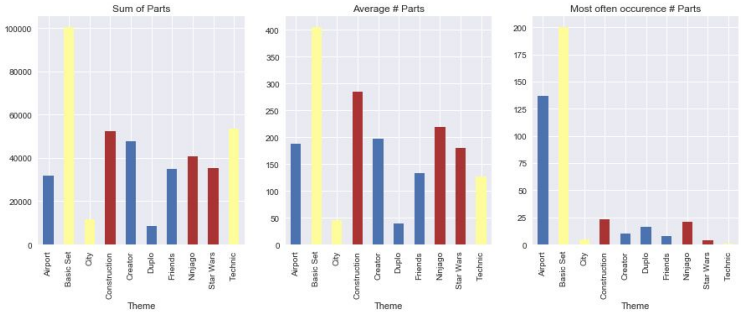

```
Accuracy: 0.683054393305
Precision: 0.690736555006
Recall: 0.683054393305
F1 Score: 0.68299843251
```

↓

```
Accuracy: 0.469665271967
Precision: 0.468978826622
Recall: 0.469665271967
F1 Score: 0.466935366146
```

Classifier Result

Possible explanation?



Most Common Year per Theme

Theme	Year
Airport	2002
Technic	2002
Creator	2004
Basic Set	2006
Construction	2009
Friends	2012
City	2014
Duplo	2014
Star Wars	2014
Ninjago	2015

Neural Net Classifier

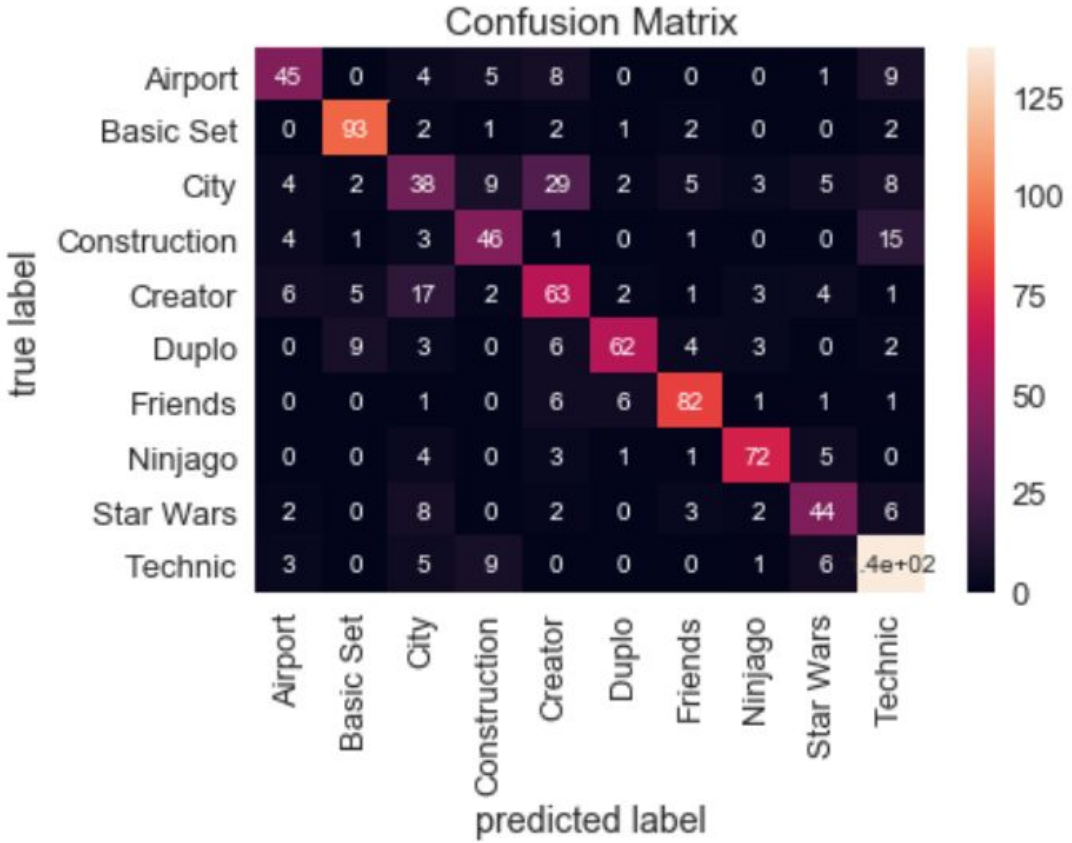
```
Accuracy: 0.713389121339
Precision: 0.711008362391
Recall: 0.713389121339
F1 Score: 0.710040276239
```

Solver: lbfgs

Maximum Number of Iterations: 400

Hidden Layers: 200 with two layers.

Confusion matrix shows that neural net had issues identifying the difference between Creator and City, as well as small issues with identifying Technic and Construction.



TEAM 6

Lean, Marc, Muhannad, Tran, Valerie

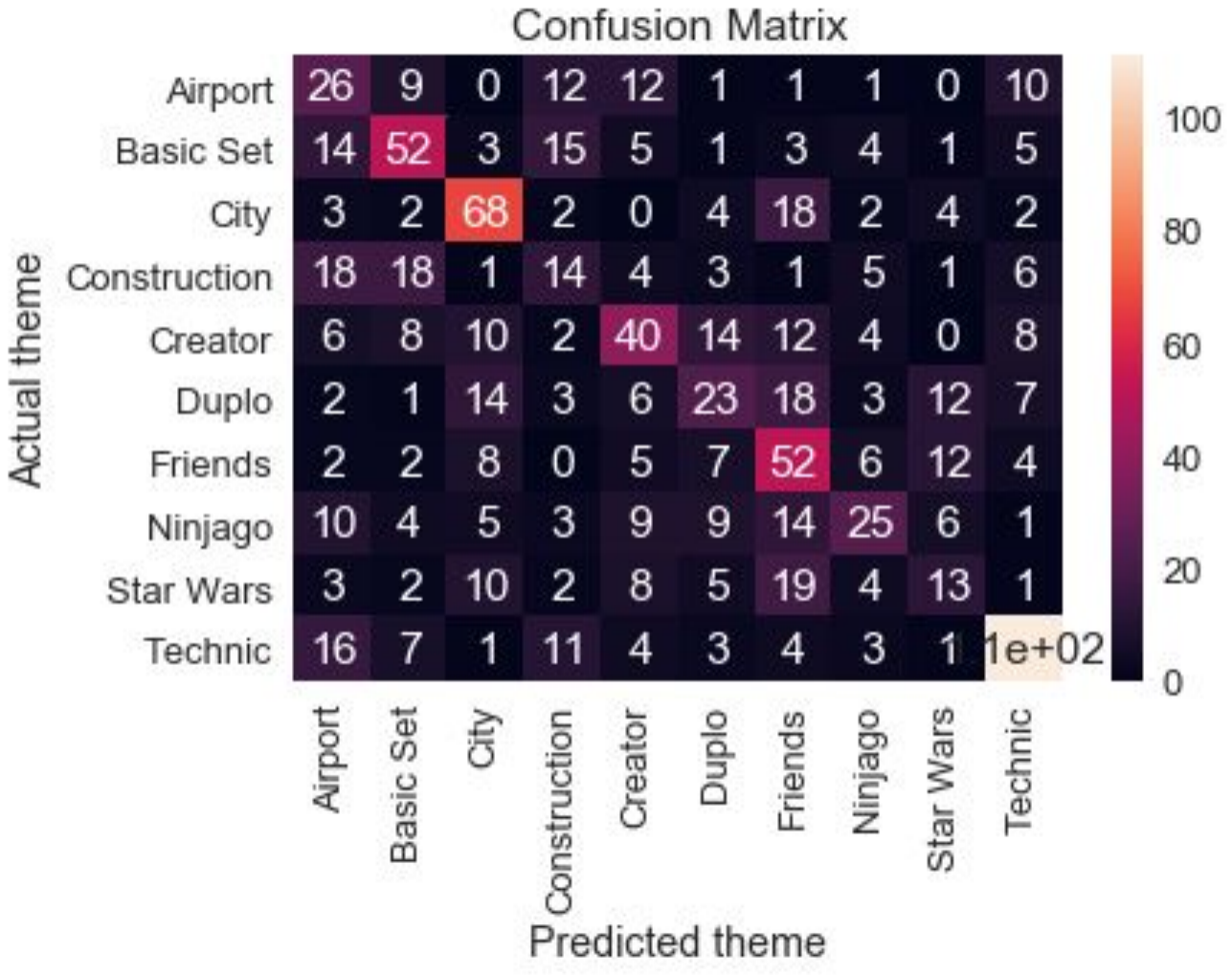
1) kNN Classifier

Predictors:

- Year
- Number of Parts
- Colors
  - [No Color]
  - Black
  - White
  - Red
  - Green
  - Blue

N = 5
Test size = 40%

Accuracy: 0.4435
Precision: 0.4446
Recall: 0.4435
F1-score: 0.4393



Interpretation

F1 Score of 0.4610 seems pretty low

Appears to work only for specific themes (City in particular)

Likely because
- [No Color]
- Black
- White
- Red
- Green
- Blue

Are only a few of many, many different possible colors

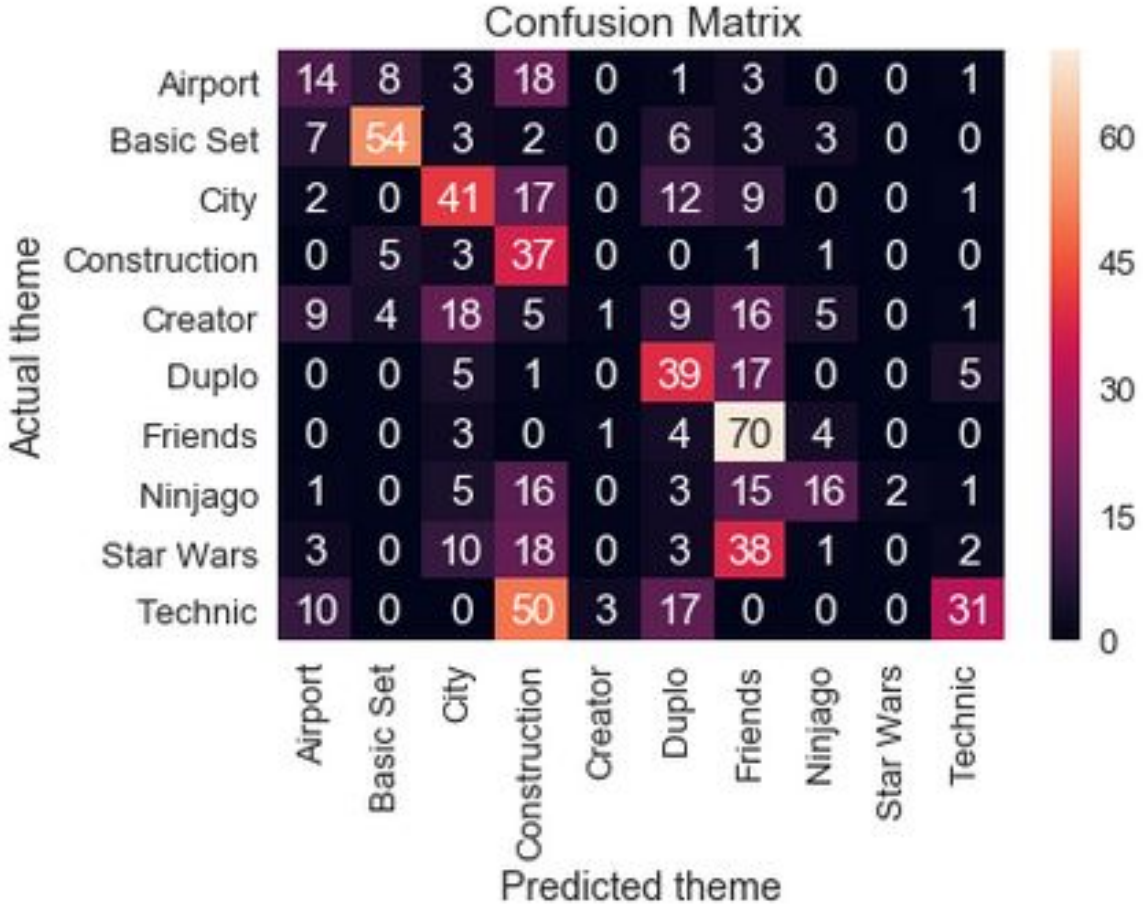
2) Naïve Bayes Classifier

Predictors:

- Year
- Number of Parts
- Colors
  - [No Color]
  - Black
  - White
  - Red
  - Green
  - Blue

Accuracy: 0.42259414225941422
Precision: 0.43186409519484731
Recall: 0.42259414225941422
F1 Score: 0.37921703788277999

- Low Scores Overall for Accuracy, Precision, Recall and F1 Score



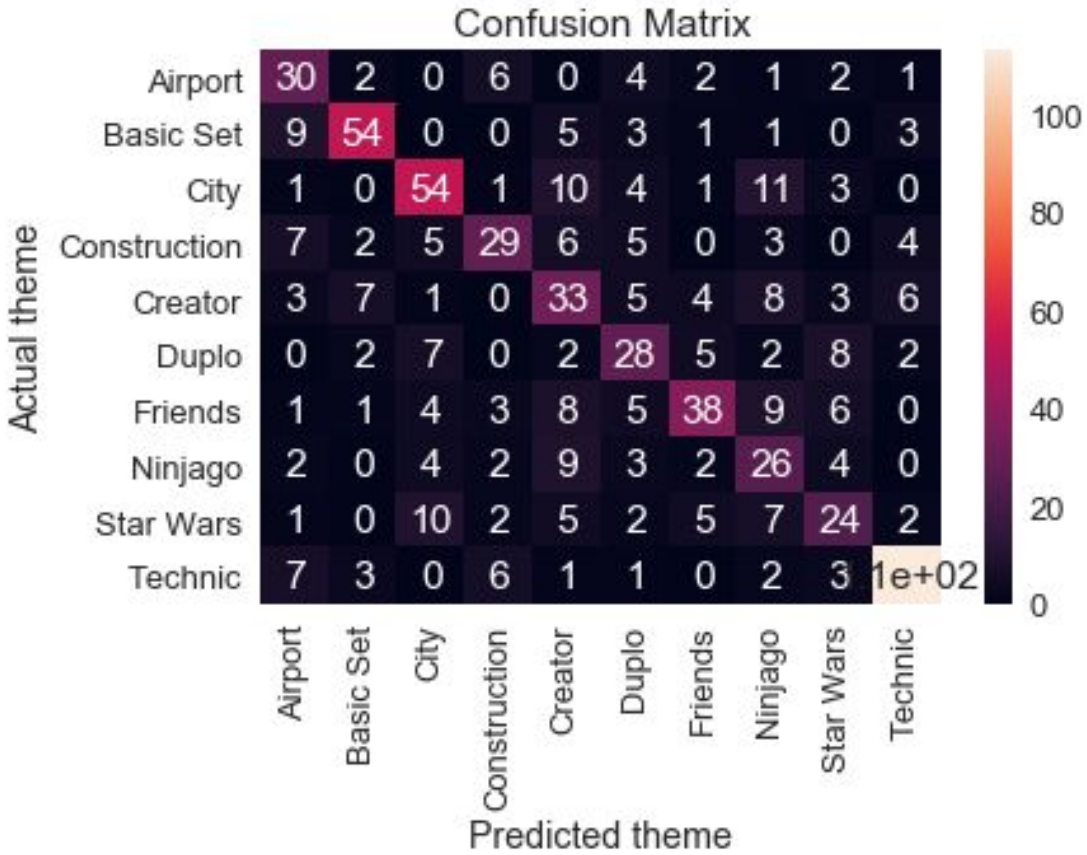
3) Decision Tree Classifier

Predictors:

- Year
- Number of Parts
- Colors
  - [No Color]
  - Black
  - White
  - Red
  - Green
  - Blue

Accuracy: 0.5983263598326359
Precision: 0.6121542532059034
Recall: 0.5983263598326359
F1 Score: 0.6020667319491237

- Everything > 50%
- Performs slightly worse than Random Forest Classifier



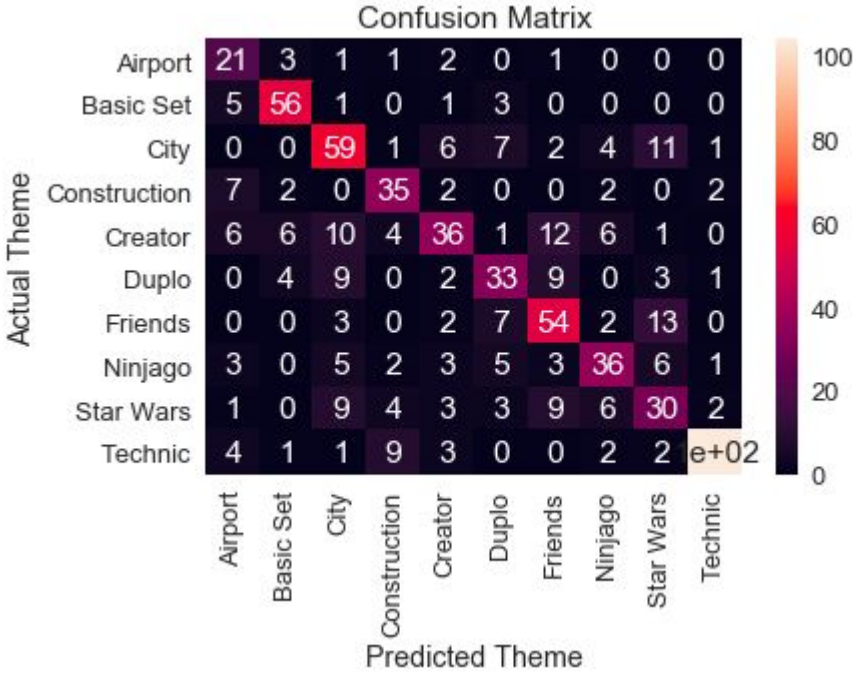
4) Random Forest Classifier

N-Estimators = 100

Parameters

- Year
- Num Parts
- Colours
  - (No Color)
  - Black
  - White
  - Red
  - Green
  - Blue

Accuracy: 0.647140864714086 49
Precision: 0.656175559184084 88
Recall: 0.647140864714086 49
F1 Score: 0.647543807028955 98



- Moderate recall and precision! → True Positive in > 50%
- Overall, reasonably poor (average) job in terms of predicting

5) Neural Network Classifier

Predictors: All the colours!
(and no other features)
I used sklearn.preprocessing StandardScaler() to normalize the data as opposed to df.div()

```
scaler = skp.StandardScaler()
# TODO: Make sure you split your data in
```

```
X = norm_lego.drop(['Theme'],axis=1)
y = norm_lego['Theme']
```

```
X_train, X_test, y_train, y_test = train_
```

```
scaler.fit(X_train)
```

```
StandardScaler(copy=True, with_mean=True,
```

```
X_train=scaler.transform(X_train)
X_test=scaler.transform(X_test)
```

```
print(skm.classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
Airport	0.47	0.69	0.56	26
Basic Set	0.85	0.81	0.83	62
City	0.56	0.56	0.56	63
Construction	0.68	0.77	0.72	39
Creator	0.49	0.43	0.46	61
Duplo	0.86	0.80	0.83	60
Friends	0.86	0.92	0.89	65
Ninjago	0.93	0.77	0.85	53
Star Wars	0.80	0.60	0.69	58
Technic	0.79	0.90	0.84	111
avg / total	0.75	0.74	0.74	598

```
#Accuracy
print(skm.accuracy_score(y_true=y_test,
                         y_pred=predictions))
```

```
0.7408026755852842
```

Accuracy	0.74
Precision	0.75
Recall	0.74
F1 Score	0.74

Confusion Matrix

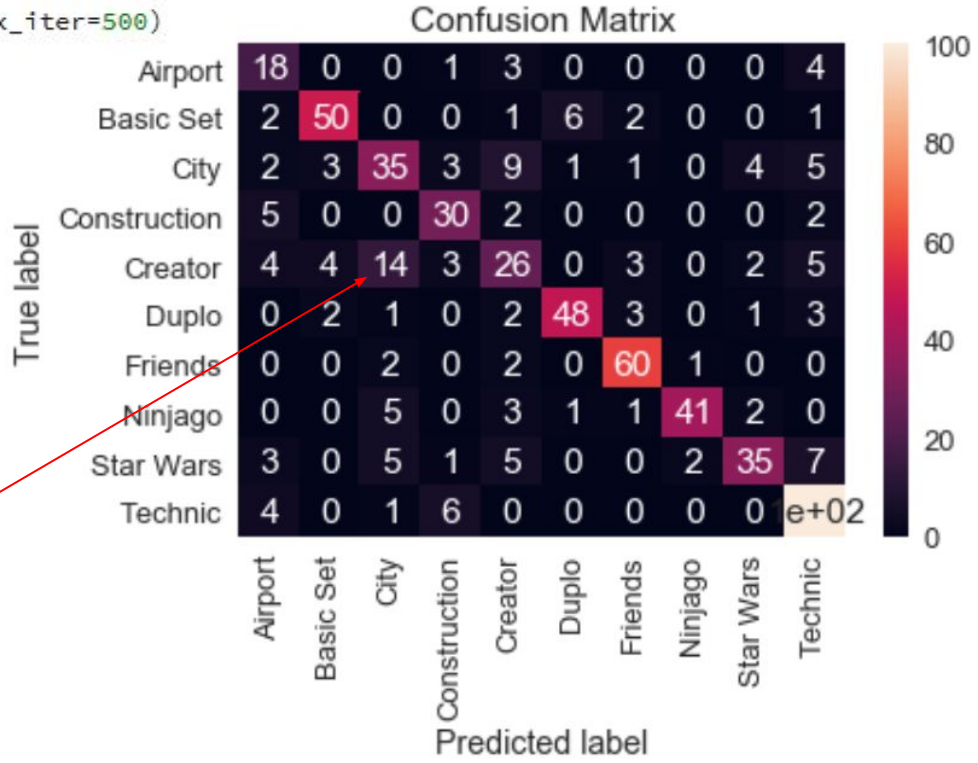
```
size= color_cols.size
mlp = MLPClassifier(hidden_layer_sizes=(size), max_iter=500)
```

After some rudimentary experimentation, it seems that it's best to stick with:

- 1 hidden layer, and
- No. of neurons = features

Otherwise, it just takes unnecessary resources or underperforms.

Interpretations:
Clearly, the classifier got it pretty well!
Creator and City themes are a little mixed up, but those are the biggest issues.



End of Slides