Artificial Intelligence and Machine Learning (CMP6087)
Group Coursework CWRK001
Faculty of Computing, Engineering and the Built Environment
Yevgeniya Kovalchuk

A Decision Tree Learning Model for Multi-class Classification of Interlocking Plastic Bricks

James Witts S14139866

16th February 2018

Summary

This report outlines coursework carried out to meet the assessment criteria for the *Artificial Intelligence (AI) and Machine Learning (ML) module CMP6087*. A machine was programmed using a decision tree algorithm for the purposes of multi-class classification of a random sample of interlocking plastic bricks.

The fictional scenario is the customer returns department of a manufacturer of toys. Processing many packages by hand often without any identifiable information about the ‘theme’ of the construction set, means inventory replenishment is difficult even though it is already known that each theme contains many different sets, and each set contains many individual parts.

Research Question

Can a machine classify a random sample of interlocking plastic bricks into themes using part colours and part names only?

Multi-Classification Algorithms: Decision Trees

The research question is a multi-classification problem because the same unit or ‘part’ can belong to multiple classifications, called ‘themes’. Decision trees are one way to solve a multi-classification problem. There are three main ways of solving a multi-class problem:

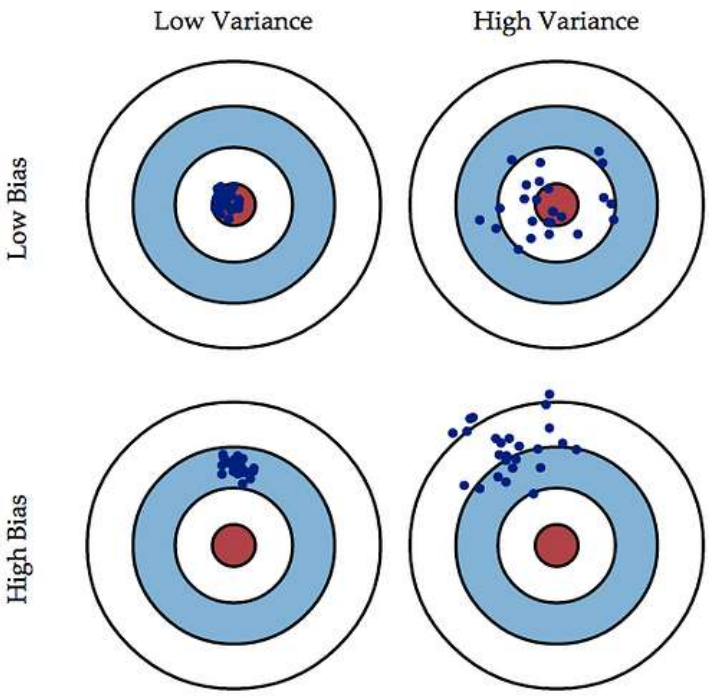
- Transformation to binary
- Extension from binary
- Hierarchical classification

Depending on the name of the part and colour of the part, the selected model must be capable of classifying toy parts into a theme.

Low variance low bias gives the best classifications (top left).

High variance, low bias leads to over-fitting, where the model is too precise and when classifying doesn't generalise well (top right).

High bias, low variance leads to under-fitting, where the model is precise but inaccurate see diagram (bottom left). (“What is bias in a machine learning algorithm?,” n.d.)



For bias related errors like under-fitting, a method called gradient boosting can be used. Gradient boosting is a technique that produces a group of weak prediction models. For this research model ten weak decision trees were used. Building trees in this stage-wise way helps the optimization of the differentiable loss function because each tree is able to make a more accurate prediction on-top of what has already been learned.

Given the gradient boosting approach seemed to be the most appropriate method it is what is being used to develop this AI.

Gradient boosting using Catboost

The python library Catboost is one library made specifically to provide this feature.

Originally for making a decision tree, splits would be made using the measures of entropy and the gini coefficient. But for libraries like Catboost and XGBoost the gradient and the hessian of the custom objective function are used. For Catboost there are nine different multi-classification algorithms for comparison. Currently “MultiClass” is the one being used (shown below,) another one of these loss functions available is the “MultiClassOneVsAll” function.

$$\frac{\sum_{i=1}^{N} w_i \log\left(\frac{e^{a_{it_i}}}{\sum_{i=0}^{M-1} e^{a_{ij}}}\right)}{\sum_{i=1}^{N} w_i},$$

$$t \in \{0, ..., M-1\}$$

(“CatBoost — Training parameters — Yandex Technologies,” n.d.)

During training, these set of ten decision trees are built one after the other. Each tree is built with reduced loss compared to the previous tree, improving the models accuracy each time. Also, there is an over-fitting detector by default which prevents trees being built when it is triggered.

The building stages for every single tree are as follows:

1. Preliminary calculation of splits or (Binarization) (“CatBoost — Binarization — Yandex Technologies,” n.d.)
2. Change of categorical features to numerical features
3. Choosing the tree structure
4. Calculating values in leaves

The most important input parameter to consider when training for speed and simplicity is the maximum depth of the tree. If tree depth is increased by one, then the leaves of the tree, the time of execution and the complexity of the algorithm approximately double. A technique sometimes used to prevent this is ‘regularization’. (PyData, n.d.)

Using Catboost for Categorical Data

Categorical values can have one possible value, for example the name of the part can be any of the following: ‘Plate 2 x 2’, ‘Minifig Knitted Cap’, ‘Brick 1 x 2’), these just a few examples. The colour of the part could be: ‘Black’, ‘White’ to name a few.

In Catboost, before each split is selected in the tree, categorical values are transformed to numeric values. This is done using various statistics on combinations of categorical values. For multi-classifications the label values are integer values starting from “0” and then these calculations are made.

Possible uses of this feature

Assuming one part “Brick 1 x 2” in the training set belongs to two different themes,“Basic Set” and “Lego City”, CatBoost can create a new theme if needed that is a combination of these two “Basic Set Lego City” themes. Approaching the problem in this way could be a useful method for inventory replenishment. Toy parts (like “Brick 1 x 2”) could be taken from the combined theme “Basic Set Lego City” and placed into either “Basic Set” or “Lego City” depending which set needs it the most.

(“CatBoost — Transforming categorical features to numerical features — Yandex Technologies,” n.d.)

Dataset description and pre-processing

The data was sourced from an online resource called ‘Rebrickable’ that offers data on official and custom construction sets. It also provides an Application Programming Interface (API) available for download via Kaggle (“LEGO Database,” n.d.). Rebrickable states in the API terms of use that all data can ‘may be used for any purpose, including commercial’ (“Terms Of Service | Rebrickable - Build with LEGO,” n.d.).

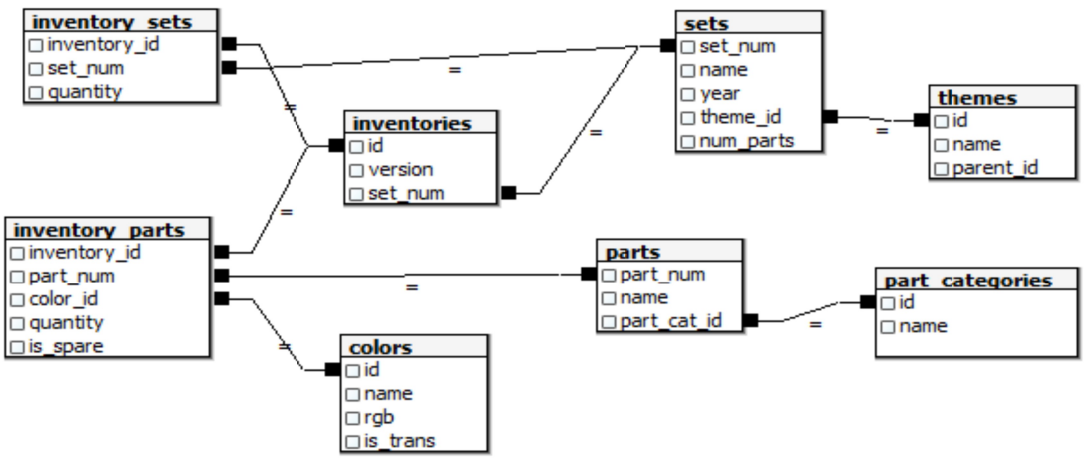
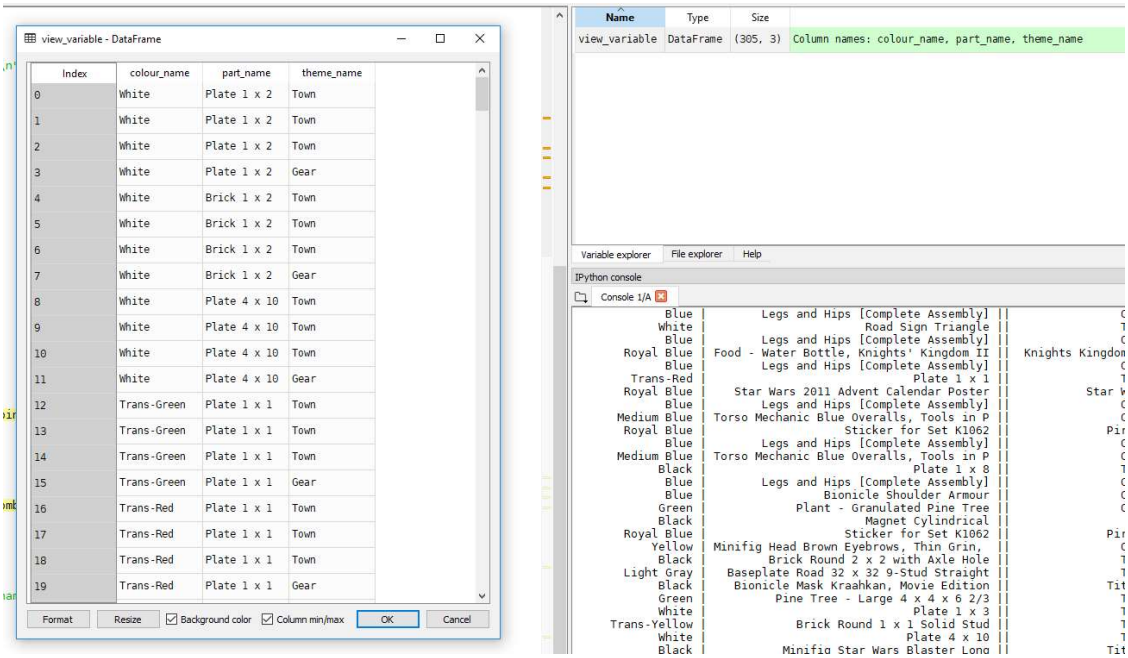


Illustration 1: Diagram of original dataset structure

The original dataset is made up of eight different comma separated value (CSV) files as shown (above). Since not all the columns were relevant to the proposed analysis using a supervised learning algorithm, merging the CSV files together left three columns while

five were dropped. The three remaining columns were then renamed to use in a *Pandas* python library object or ‘dataframe’ containing twenty five thousand, nine hundred and ninety three parts in total. See the screenshot for the resulting dataframe (below).



2. Illustration: List of features/variables being considered after combining all three csv files into one python object property with type pandas data frame.

The “theme_name” column was not used in the test dataset when the random splitting using the python ‘test train split’ function was completed because this was the unknown label the research question aimed to identify. When the main script (see Appendix III: Crk_Main_s14139866.py) was run, the initial output to screen was as follows (see Appendix V: Console Outputs):

```
There are this many bricks: 25993
There are this many unique names: 25779
There are this many unique themes: 402
There are this many unique colours: 135
```

The output above gives a broad overview of the contents of the original dataset. Then came making the dataset more usable for the purposes of testing the model (see Appendix II: Crk_Catboost.py), this meant merging and dropping columns and the removal of parts with the value of “[NoColour]” or “Unknown” for the column “colour_name”. Which results in the following output:

```
There are this many bricks: 305
There are this many unique names: 61
There are this many unique themes: 27
There are this many unique colours: 18
```

The program (See Appendix V: Console Outputs) also produced the following visualisations using the python *matplotlib* library (“Matplotlib: Python plotting — Matplotlib 2.2.2 documentation,” n.d.).

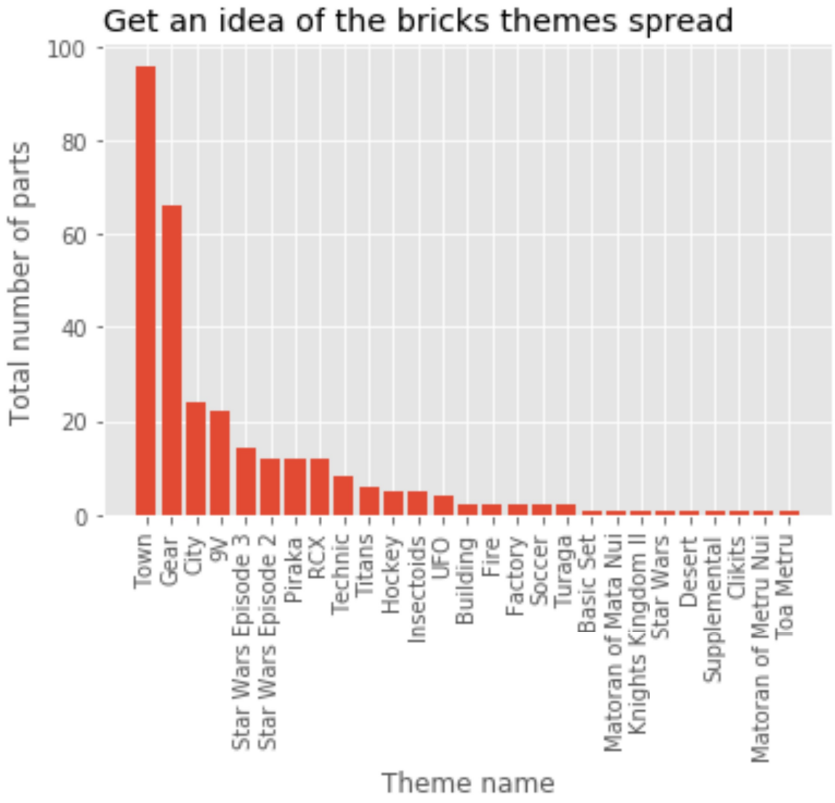


Illustration 3: Range of themes in final dataset

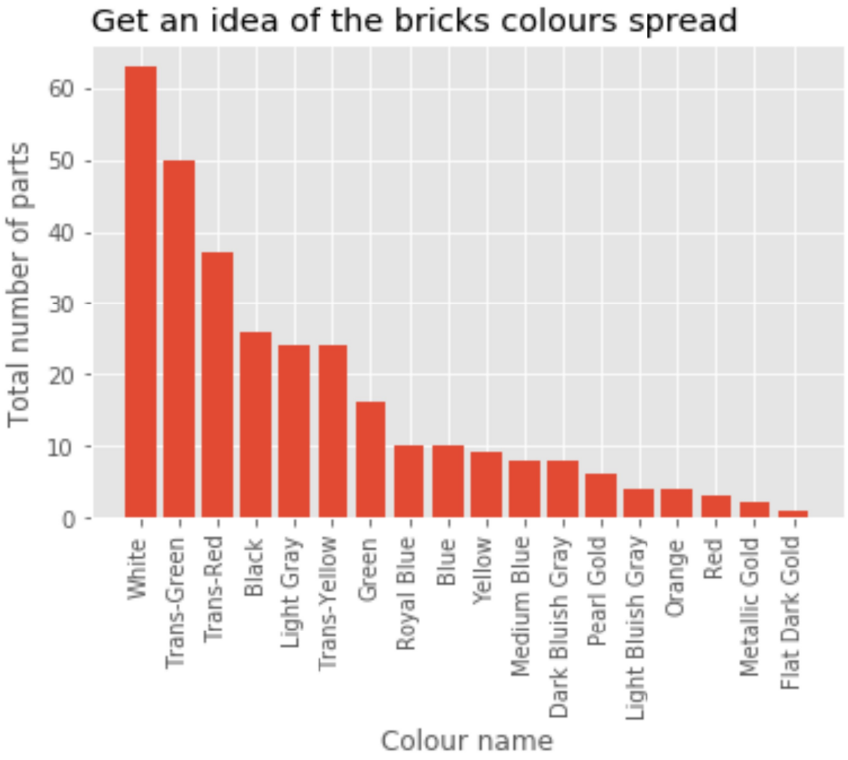


Illustration 4: Range of colours in final dataset

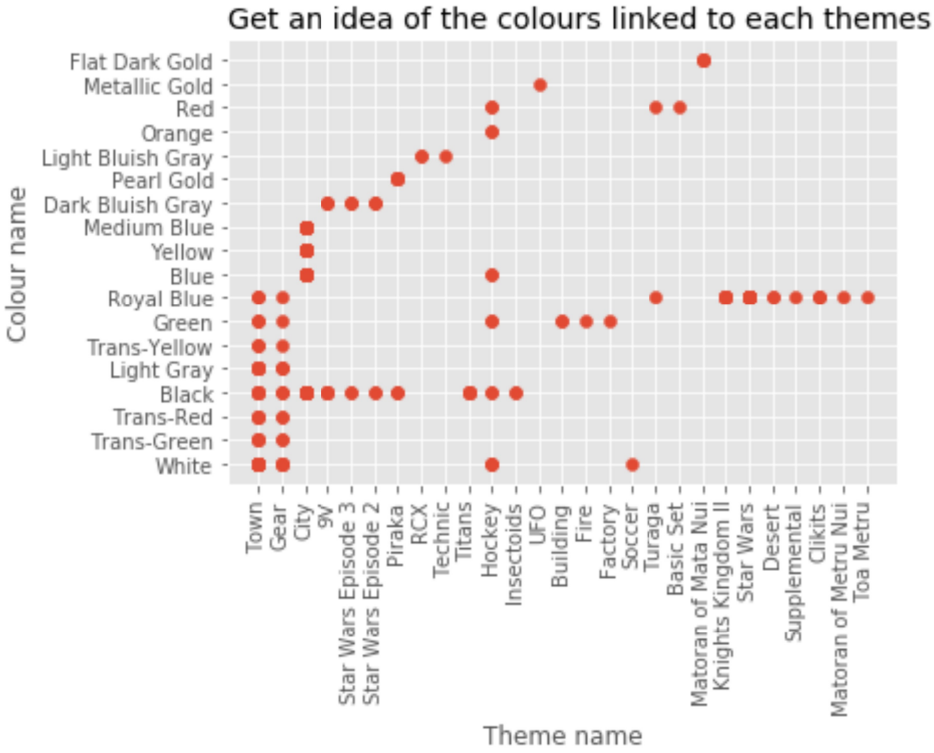


Illustration 5: Themes and their respective colours. Used to verify the test data

These results seemed sufficient for the purpose of splitting into test and training data.

Splitting was achieved by adding the “splitIntoTrainTest” function to the program as follows (See Appendix I: Crk_Handle_Dataset.py):

```
def splitIntoTrainTest(self):
 self.train_df, self.test_df = train_test_split(self.df_combined,
test_size=0.2)
```

The output during the training split for this specific Catboost instance was as follows:

```
This many bricks have been used in training: 244
This many themes out of 27 are used in this training instance: 25
```

Only twenty five of the possible twenty seven themes were selected for the supervised learning. This was due to the random split of the dataframe. It is unknown how this type of split might effect the final test results. For example, it may cause problems in accurately classifying those themes that haven't been used in training.

At the end of the data pre-processing phase the training dataset being output indicates that its a reasonable candidate for use in an AI Model and the test dataset is suitable for the testing of the supervised model.

Results and discussion

Here are some of the results from Appendix V: Console Outputs Catboost testing section:

Tested colour	Tested part name	Classification/ theme	Probability of being accurate.	Human check for the actual theme.
Black	Bionicle Shoulder Armour	Hockey	61 percent	True: 2 entries both in ‘Hockey’ theme
Black	Minifig Knitted Cap	City	98 percent	True: 24 entries all in ‘City’ theme
Blue	Legs and Hips [Complete Assembly]	City	100 percent	True: 24 entries all in ‘City’ theme
White	Sports Hockey Chest Protector	Town	80 percent	False: 2 entries both in ‘Hockey’ theme
Light Gray	Baseplate Road 32 x 32 9-Stud Straight	Town	96 percent	In-between: 4 entries total, 3 of them in ‘Town’ one of them in ‘Gear’ theme.

On average the results show that parts have been successfully classified into their respective themes, along with information from Catboost on the probability of its own accuracy and just for the cases above a human check has also been done which shows that the reliability of these catboost percentages isn’t great. There are two interesting cases shown above where one part belongs to two themes and the current AI is only able to classify it into to one of them. The other significant case is where the AI is eighty percent sure its accurate but is actually in-accurate.

Visualisations of the Catboost decision tree model are not currently available but are being worked on. XGBoost can plot a decision tree model but it doesn’t provide any helpful understanding yet since it uses the encoded values when printed to screen.

Conclusion and future developments

Overall this AI has shown to be a very successful aid to classifying toy parts into their themes. With more data, accuracy would be improved and could potentially replace the need for any human classification of toy parts.

For future developments the AI could be programmed to provide a better user interface and more statistics on a prompt to give better oversight of the classification process.

References

CatBoost — Binarization — Yandex Technologies [WWW Document], n.d. URL https://tech.yandex.com/catboost/doc/dg/concepts/binarization-docpage/ (accessed 6.4.18).

CatBoost — Training parameters — Yandex Technologies [WWW Document], n.d. URL https://tech.yandex.com/catboost/doc/dg/concepts/python-reference_parameters-list-docpage/ (accessed 6.4.18).

CatBoost — Transforming categorical features to numerical features — Yandex Technologies [WWW Document], n.d. URL https://tech.yandex.com/catboost/doc/dg/concepts/algorithm-main-stages_cat-to-numberic-docpage/ (accessed 6.4.18).

LEGO Database [WWW Document], n.d. URL https://www.kaggle.com/rtatman/lego-database (accessed 3.13.18).

Matplotlib: Python plotting — Matplotlib 2.2.2 documentation [WWW Document], n.d. URL https://matplotlib.org/ (accessed 6.7.18).

PyData, n.d. Jaroslaw Szymczak - Gradient Boosting in Practice: a deep dive into xgboost.

Terms Of Service | Rebrickable - Build with LEGO [WWW Document], n.d. URL https://rebrickable.com/terms/ (accessed 4.3.18).

What is bias in a machine learning algorithm? [WWW Document], n.d. . Data Sci. Anal. Big Data Discuss. URL https://discuss.analyticsvidhya.com/t/what-is-bias-in-a-machine-learning-algorithm/2171 (accessed 6.4.18).

Appendix I: Crk_Handle_Dataset.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 28 19:15:13 2018
@author: jay
"""
import pandas as pd
from sklearn.model_selection import train_test_split
class Crk_Handle_Dataset():
    def __init__(self):
        self.df_colours = None
        self.df_inventories = None
        self.df_inventory_parts = None
        self.df_inventory_sets = None
        self.df_part_categories = None
        self.df_parts = None
        self.df_sets = None
        self.df_themes = None
        self.train_df = None
        self.test_df = None
        # Step 1.0
        # Import the data from the 8 csv files into data frames.
        self.importDataset()
        # Step 1.1
        # Prepare the data frame columns to be merged.
        self.prepareForMerge()
        # Step 2.0
        # Make a combined inventory of parts with all information
from csv's...
        # ...using the schema in the image file.
        self.mergeAllColumns()
        # Step 3.0
        # Drop unwanted columns
        self.dropUnwantedColumns()
        # Step 4.0
        # Remove anything with missing colour or colour unknown
        self.dropUnclassifiableData()
        # Step 5.0
        # Split into 80% train 20% test data.
        self.splitIntoTrainTest()
    def importDataset(self):
        self.df_colours = pd.read_csv('lego-database/colors.csv')
        self.df_inventories =
pd.read_csv('lego-database/inventories.csv')
        self.df_inventory_parts =
pd.read_csv('lego-database/inventory_parts.csv')
        self.df_inventory_sets =
pd.read_csv('lego-database/inventory_sets.csv')
        self.df_part_categories =
pd.read_csv('lego-database/part_categories.csv')
        self.df_parts = pd.read_csv('lego-database/parts.csv')
        self.df_sets = pd.read_csv('lego-database/sets.csv')
        self.df_themes = pd.read_csv('lego-database/themes.csv')
    def prepareForMerge(self):
        self.df_colours.rename(columns={'id': 'colour_id'},
inplace=True)
```

```
        self.df_colours.rename(columns={'name': 'colour_name'},
inplace=True)
        self.df_inventories.rename(columns={'id': 'inventory_id'},
inplace=True)
        self.df_inventory_parts.rename(columns={'quantity':
'quantity_parts'}, inplace=True)
        self.df_inventory_parts.rename(columns={'color_id':
'colour_id'}, inplace=True)
        self.df_inventory_sets.rename(columns={'quantity':
'quantity_sets'}, inplace=True)
        self.df_part_categories.rename(columns={'id':
'part_cat_id'}, inplace=True)
        self.df_part_categories.rename(columns={'name':
'part_cat_name'}, inplace=True)
        self.df_parts.rename(columns={'name': 'part_name'},
inplace=True)
        self.df_sets.rename(columns={'name': 'set_name'},
inplace=True)
        self.df_themes.rename(columns={'id': 'theme_id'},
inplace=True)
        self.df_themes.rename(columns={'name': 'theme_name'},
inplace=True)
    def mergeAllColumns(self):
        # df_inventory_parts <- df_colours
        # df_inventory_parts <- df_parts <- df_part_categories
        # df_inventory_parts <- df_inventories <-
df_inventory_sets <- df_sets <- df_themes
        ### merge path 1 ###
        self.df_combined = pd.merge(self.df_inventory_parts,
self.df_colours, on='colour_id')
        ### merge path 2 ###
        df_merge_2 = pd.merge(self.df_parts,
self.df_part_categories, on='part_cat_id')
        self.df_combined = pd.merge(self.df_combined, df_merge_2,
on='part_num')
        ### merge path 3 ###
        df_merge_3_1 = pd.merge(self.df_sets, self.df_themes,
on='theme_id')
        df_merge_3_2 = pd.merge(self.df_inventory_sets,
df_merge_3_1, on='set_num')
        self.df_combined = pd.merge(self.df_combined,
df_merge_3_2, on='inventory_id')
    def dropUnwantedColumns(self):
        self.df_combined = self.df_combined.drop(['part_num',
'year', 'inventory_id',

'colour_id','set_num','is_spare',

'quantity_parts','quantity_sets','num_parts',

'parent_id','part_cat_name','part_cat_id'
                                    ,'theme_id', 'rgb',
'is_trans',
                                    'set_name'], axis=1)
    def dropUnclassifiableData(self):
        self.df_combined =
self.df_combined[self.df_combined.colour_name != "[No Color]"]
        self.df_combined =
self.df_combined[self.df_combined.colour_name != "Unknown"]
```

```
    def splitIntoTrainTest(self):
        self.train_df, self.test_df =
train_test_split(self.df_combined, test_size=0.2)
```

Appendix II: Crk_Catboost.py

```
"""
Train a decision tree to accurately classify which 'theme' a
'part'
belongs to just by its colour and its name.
"""
import numpy as np
import pandas as pd
from catboost import Pool, CatBoostClassifier

class J_Catboost_Model():
    def __init__(self, train_df, test_df):
        self.train_df = train_df
        self.test_df = test_df
        self.train_values = None
        self.train_labels = None
        self.train_labels_mapper = None
        self.train_values_cat_index = None
        self.train_pool = None
        self.test_values = None
        self.testvalues_cat_index = None
        self.test_pool = None
        self.clf = None
        self.preds_class = None
        self.preds_class_str = None
        self.preds_prob = None
        # Step 5.1
        # Make a train pool object for CatBoostClassifier
        self.makePoolTrain()
        # Step 5.2
        # Make a test pool object for CatBoostClassifier
        self.makePoolTest()
        # Step 6.0
        # Initialize CatBoostClassifier
        self.initModel()
        # Step 7.0
        # Fit the training data into the model.
        self.fitModel()
        # Step 8.0
        # make the prediction using the resulting model
        self.accuracyTest()
    def makePoolTrain(self):
        self.train_values = self.train_df.drop(['theme_name'],
axis=1)
        self.train_labels =
self.train_df.theme_name.astype('category').cat.codes
        self.train_labels_mapper =
dict(enumerate(self.train_df['theme_name'].astype('category').cat.
categories))
        self.train_values_cat_index =
np.where(self.train_values.dtypes != np.float)[0]
        self.train_pool =
Pool(self.train_values,self.train_labels,cat_features=self.train_v
alues_cat_index)
    def makePoolTest(self):
        self.test_values = self.test_df.drop(['theme_name'],
axis=1)
```

```
        self.test_values_cat_index =
np.where(self.test_values.dtypes != np.float)[0]
        self.test_pool =
Pool(self.test_values,cat_features=self.test_values_cat_index)
    def initModel(self):
        self.clf = CatBoostClassifier(
            iterations=10,#how many trees
            learning_rate=1,
            depth=6,#depth of each tree

classes_count=len(self.train_labels.unique().tolist()),
            loss_function='MultiClass',
            logging_level='Verbose'
        )
    def fitModel(self):
            self.clf.fit(self.train_pool, plot=True)
            # To see the training process plotted it needs to run
in a jypyter notebook.
            # This can be done by starting the server on localhost
and making a notebook.
            # Open another terminal and run: jupyter nbextension
enable --py widgetsnbextension
    def accuracyTest(self):
        self.preds_class = self.clf.predict(self.test_pool,
prediction_type='Class')
        self.preds_class_str =
pd.DataFrame(self.preds_class.astype(int), columns=['theme'])
        self.preds_class_str =
self.preds_class_str['theme'].map(self.train_labels_mapper).astype
('category')
        self.preds_prob = self.clf.predict(self.test_pool,
prediction_type='Probability')
        # The preds_prob is a matrix of probabilities for each
preds_class element,
        # so i only need to show the max probability in that
matrix.
        # (the highest probability index also happens to be the
preds_class value)
```

Appendix III: Crk_Main_s14139866.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 28 19:21:44 2018

@author: jay
"""

from Crk_Handle_Dataset import Crk_Handle_Dataset
from Crk_Catboost import J_Catboost_Model
from Crk_XGBoost import J_XGBoost_Model

import matplotlib.pyplot as plt
import ipywidgets
import widgetsnbextension
#from catboost import CatboostIpythonWidget
from IPython.display import display
from IPython.core.display import HTML

data_obj = Crk_Handle_Dataset()
catboost_obj =
J_Catboost_Model(data_obj.train_df,data_obj.test_df)
#xgboost_obj = J_XGBoost_Model(data_obj.train_df,data_obj.test_df)

# Step 9.0
# Print results
def printCatboost(catboost_obj):
    count = 0
    print("\n"*2)
    print("===== START OF S14139866 LEGO BRICKS AI ===== \n"
          "N.B. Please adjust window width to fit formatting of
100 characers.\n\n"
          "=== Understanding the Data === \n"
          "From the 8 csv files... \n"
          "There are this many bricks: %d \n"
          "There are this many unique names: %d \n"
          "There are this many unique themes: %d \n"
          "There are this many unique colours: %d \n"
          %(len(data_obj.df_parts),
            len(data_obj.df_parts['part_name'].unique().tolist()),

len(data_obj.df_themes['theme_name'].unique().tolist()),

len(data_obj.df_colours['colour_name'].unique().tolist())))
    print("After merging, dropping and removal of parts \n"
          "with '[No Colour]' and 'Unknown' for the column
'colour_name' \n"
          "the data now looks like this: \n"
          "There are this many bricks: %d \n"
          "There are this many unique names: %d \n"
          "There are this many unique themes: %d \n"
          "There are this many unique colours: %d \n\n\n"
          %(len(data_obj.df_combined),

len(data_obj.df_combined['part_name'].unique().tolist()),
```

```
len(data_obj.df_combined['theme_name'].unique().tolist()),

len(data_obj.df_combined['colour_name'].unique().tolist())))

    plt.style.use('ggplot')
    fig = plt.figure(1)
    plt.title('Get an idea of the bricks themes spread',
loc='left')
    ax = fig.add_subplot(1,1,1)
    ax.bar(data_obj.df_combined['theme_name'].unique().tolist(),
data_obj.df_combined['theme_name'].value_counts())
    plt.xticks(rotation=90)
    ax.set_ylabel('Total number of parts')
    ax.set_xlabel('Theme name')
    fig2 = plt.figure(2)
    plt.title('Get an idea of the bricks colours spread',
loc='left')
    ax2 = fig2.add_subplot(1,1,1)
    ax2.bar(data_obj.df_combined['colour_name'].unique().tolist(),
data_obj.df_combined['colour_name'].value_counts())
    plt.xticks(rotation=90)
    ax2.set_ylabel('Total number of parts')
    ax2.set_xlabel('Colour name')
    fig3 = plt.figure(3)
    plt.title('Get an idea of the colours linked to each themes',
loc='left')
    ax3 = fig3.add_subplot(1,1,1)

ax3.scatter(data_obj.df_combined['theme_name'],data_obj.df_combine
d['colour_name'])
    plt.xticks(rotation=90)
    ax3.set_ylabel('Colour name')
    ax3.set_xlabel('Theme name')
    ###
    plt.show()
    print("--> This is helpful for checking the tested data
later \n"
          "used as a rough estimate to see if the program is
working \n"
          "as expected. \n")

    print("=== Catboost Model Training ===")
    print("This many bricks have been used in training: %d \n"
          "How many themes out of %d"
          " are used in this training instance: %d \n\n"
          %(len(catboost_obj.train_df),

len(data_obj.df_combined['theme_name'].unique().tolist()),
            len(catboost_obj.train_labels.unique().tolist())))
    print("=== Catboost Model Testing ===")
    test_values_colour_list =
catboost_obj.test_values['colour_name'].tolist()
    test_values_partname_list =
catboost_obj.test_values['part_name'].tolist()
    print("This many bricks have been classified with their
probability: %d"
          %(len(catboost_obj.test_df)))
    print("_"*100)
```

```
    print ("%17s | %40s ||%19s | %3s" % ('Tested colour', 'Tested
part name','Classification/theme', 'Probability'))
    print("_"*100)
    for j in catboost_obj.preds_class:
        var = int(catboost_obj.preds_class[count])
        print ("%17s | %40.40s || %19s | %3.2f" %
(test_values_colour_list[count],
        test_values_partname_list[count],
catboost_obj.preds_class_str[count],
catboost_obj.preds_prob[count][var]))
        count=count+1
    print("\n"*2)
    print("The probability for each theme for every tested part
(27x61): \n")
    for p in catboost_obj.preds_prob:
        for u in p:
            print ( end='' "%2.2f |" % (u))
        print("\n")
    print("Please adjust window width to fit formatting of 100
characers.")
    print("===== END OF S14139866 LEGO BRICKS AI =====")

view_variable = data_obj.df_combined

printCatboost(catboost_obj)
```

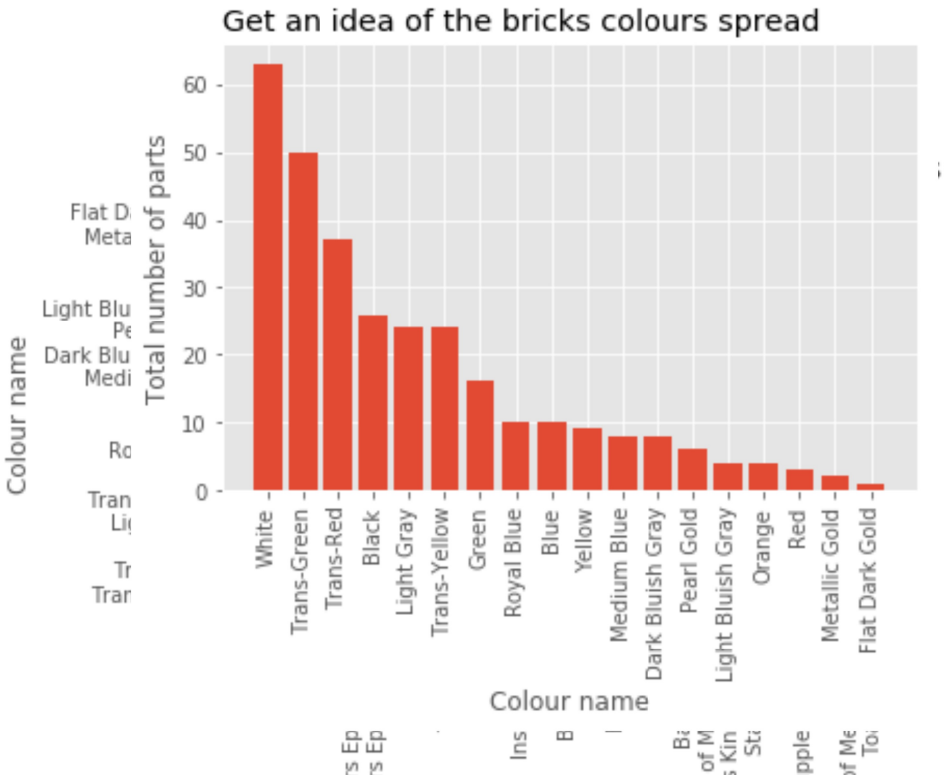
Appendix IV: Crk_XGBoost.py

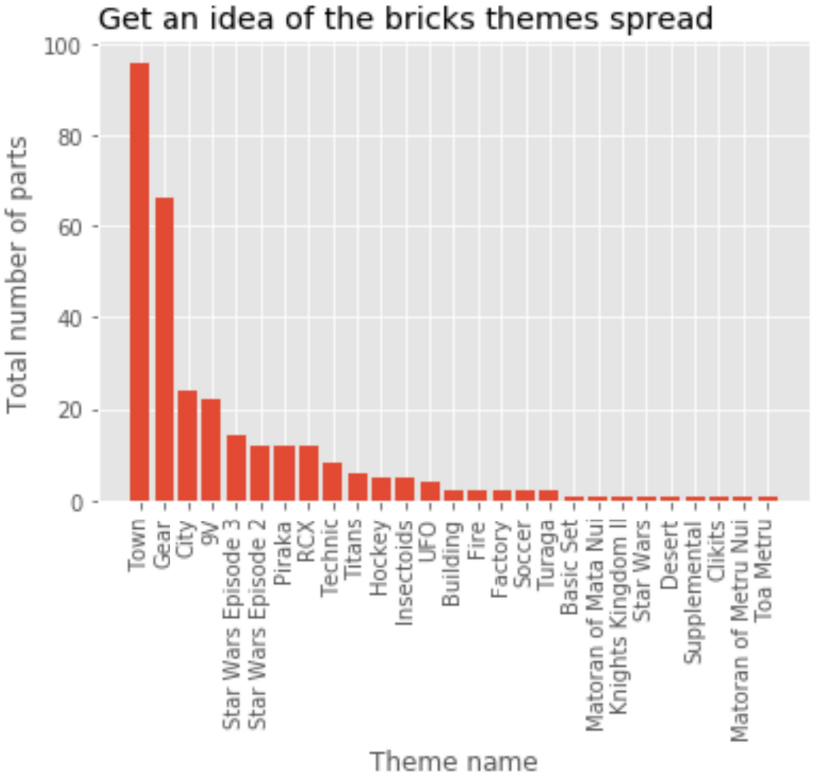
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 28 19:12:20 2018
@author: j
"""
import xgboost as xgb
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
import pandas as pd
import numpy as np

class J_XGBoost_Model():
    def __init__(self, train_df, test_df):
        self.train_df = train_df
        self.test_df = test_df
        self.model = xgb.XGBClassifier()
        self.dummies_train_x, self.dummies_train_y =
self.encodeData(self.train_df)
        self.dummies_test_x, self.dummies_train_y =
self.encodeData(self.test_df)
        self.model.fit(self.dummies_train_x, self.dummies_train_y)
    def encodeData(self, pandas_frame):
        dataset = pandas_frame.values
        # split data into X and y, there are 3 colums to split.
        X = dataset[:,0:2]
        X = X.astype(str)
        Y = dataset[:,2]
        # encode string input values as integers
        encoded_x = None
        for i in range(0, X.shape[1]):
            label_encoder = LabelEncoder()
            feature = label_encoder.fit_transform(X[:,i])
            feature = feature.reshape(X.shape[0], 1)
            onehot_encoder = OneHotEncoder(sparse=False)
            feature = onehot_encoder.fit_transform(feature)
            if encoded_x is None:
                encoded_x = feature
            else:
                encoded_x = np.concatenate((encoded_x, feature),
axis=1)
        # encode string class values as integers
        label_encoder = LabelEncoder()
        label_encoder = label_encoder.fit(Y)
        label_encoded_y = label_encoder.transform(Y)
        return encoded_x, label_encoded_y
```

Appendix V: Console Outputs

```
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct  6 2017,
12:04:38)
Type "copyright", "credits" or "license" for more information.
IPython 6.1.0 -- An enhanced Interactive Python.
Restarting kernel...
runfile('/Users/jay/Desktop/MachineLearning/MachineLearningCwrk/
Crk_Main_s14139866.py',
wdir='/Users/jay/Desktop/MachineLearning/MachineLearningCwrk')
<IPython.core.display.HTML object>
0:      learn: -1.6951822       total: 386ms    remaining: 3.47s
A Jupyter Widget
1:      learn: -1.3669964       total: 976ms    remaining: 3.9s
2:      learn: -1.1587416       total: 1.58s    remaining: 3.68s
3:      learn: -1.0564920       total: 2.16s    remaining: 3.25s
4:      learn: -1.0188823       total: 2.28s    remaining: 2.28s
5:      learn: -0.9453258       total: 2.81s    remaining: 1.87s
6:      learn: -0.8719993       total: 3.35s    remaining: 1.44s
7:      learn: -0.8168341       total: 3.89s    remaining: 972ms
8:      learn: -0.7465696       total: 4.42s    remaining: 491ms
9:      learn: -0.6748184       total: 5.04s    remaining: 0us

===== START OF S14139866 LEGO BRICKS AI =====
N.B. Please adjust window width to fit formatting of 100
characers.

=== Understanding the Data ===
From the 8 csv files...
There are this many bricks: 25993
There are this many unique names: 25779
There are this many unique themes: 402
There are this many unique colours: 135

After merging, dropping and removal of parts with '[No Colour]'
and 'Unknown' for the column 'colour_name'  the data now looks
like this:

There are this many bricks: 305
There are this many unique names: 61
There are this many unique themes: 27
There are this many unique colours: 18

/Users/jay/anaconda3/lib/python3.6/site-packages/matplotlib/cbook/
deprecation.py:106: MatplotlibDeprecationWarning: Adding an axes
using the same arguments as a previous axes currently reuses the
earlier instance.

warnings.warn(message, mplDeprecation, stacklevel=1)
```

NOTE: In a future version, a new instance will always be created and returned. Meanwhile, this warning can be suppressed, and the future behavior ensured, by passing a unique label to each axes instance.


Get an idea of the bricks colours spread
Total number of parts
Colour name
0
10
20
30
40
50
60
White
Trans-Green
Trans-Red
Black
Light Gray
Trans-Yellow
Green
Royal Blue
Blue
Yellow
Medium Blue
Dark Bluish Gray
Pearl Gold
Light Bluish Gray
Orange
Red
Metallic Gold
Flat Dark Gold


Get an idea of the bricks themes spread
Total number of parts
Theme name
0
20
40
60
80
100
Town
Gear
City
9V
Star Wars Episode 3
Star Wars Episode 2
Piraka
RCX
Technic
Titans
Hockey
Insectoids
UFO
Building
Fire
Factory
Soccer
Turaga
Basic Set
Matoran of Mata Nui
Knights Kingdom II
Star Wars
Desert
Supplemental
Clikits
Matoran of Metru Nui
Toa Metru

--> This is helpful for checking the tested data later used as a rough estimate to see if the program is working as expected.

```
=== Catboost Model Training ===

This many bricks have been used in training: 244
How many themes out of 27 are used in this training instance: 25

=== Catboost Model Testing ===
This many bricks have been classified with their probability: 61
```

Tested colour	Tested part name	Classification/theme	Probability
Black	Bionicle Shoulder Armour	Hockey	0.61
Black	Minifig Knitted Cap	City	0.98
Blue	Legs and Hips [Complete Assembly]	City	1.00
White	Sports Hockey Chest Protector	Town	0.80
Light Gray	Baseplate Road 32 x 32 9-Stud Straight	Town	0.96
Black	Plate 1 x 8	Gear	0.48
Light Gray	Minifig Lance	Town	0.71
Red	Bionicle Mask Hau Nuva Poisoned - Green	Star Wars Episode 2	0.12
Royal Blue	Star Wars 2011 Advent Calendar Poster	Star Wars	0.98
Medium Blue	Torso Mechanic Blue Overalls, Tools in P	City	1.00
Black	Technic Pin Long with Friction Ridges Le	Insectoids	0.21
Trans-Red	Brick Round 1 x 1 Solid Stud	Town	0.93
Yellow	Minifig Head Brown Eyebrows, Thin Grin,	City	1.00
Black	Minifig Knitted Cap	City	0.98
White	Road Sign Round	Town	0.82
Light Gray	Baseplate Road 32 x 32 9-Stud Curve	Town	0.96
Royal Blue	Lord Vladek Sword	Knights Kingdom II	0.50
Light Gray	Baseplate Road 32 x 32 9-Stud Crossroads	Town	0.71
Trans-Green	Plate 1 x 1	Town	0.92
Blue	Legs and Hips [Complete Assembly]	City	1.00
Royal Blue	Star Wars 2011 Advent Calendar Poster	Star Wars	0.98

```
       Pearl Gold |                   Hose, Ribbed 7mm D. 6L ||
Piraka | 0.76
            Black |          Minifig Star Wars Blaster Short ||
Star Wars Episode 2 | 0.20
       Royal Blue | Food - Water Bottle, Knights' Kingdom II ||
Knights Kingdom II | 0.79
      Medium Blue | Torso Mechanic Blue Overalls, Tools in P ||
City | 1.00
            White |                                Brick 2 x 4 ||
Town | 0.52
           Yellow | Minifig Head Brown Eyebrows, Thin Grin,  ||
City | 1.00
       Royal Blue |    Star Wars 2011 Advent Calendar Poster ||
Star Wars | 0.98
       Light Gray |                             Minifig Lance ||
Town | 0.71
       Royal Blue |            Clikits Jewelry Box (3-level) ||
Clikits | 0.27
            Black |                                Plate 2 x 2 ||
Town | 0.42
            White |                           Road Sign Round ||
Town | 0.82
            Black |                      Minifig Knitted Cap ||
City | 0.98
            Black |                      Minifig Knitted Cap ||
City | 0.98
       Pearl Gold |                   Hose, Ribbed 7mm D. 6L ||
Piraka | 0.76
       Light Gray | Baseplate Road 32 x 32 9-Stud Crossroads ||
Town | 0.71
           Yellow | Minifig Head Brown Eyebrows, Thin Grin,  ||
City | 1.00
            Blue |        Legs and Hips [Complete Assembly] ||
City | 1.00
       Royal Blue |    Star Wars 2011 Advent Calendar Poster ||
Star Wars | 0.98
            Blue |        Legs and Hips [Complete Assembly] ||
City | 1.00
            Black |                      Minifig Knitted Cap ||
City | 0.98
       Royal Blue | Knights' Kingdom II: The Quest for the H ||
Knights Kingdom II | 0.50
 Dark Bluish Gray |                   Train Track  9V Curved ||
9V | 0.62
           Yellow | Minifig Head Brown Eyebrows, Thin Grin,  ||
City | 1.00
            Black |                                Plate 2 x 2 ||
Town | 0.42
       Royal Blue |    Star Wars 2011 Advent Calendar Poster ||
Star Wars | 0.98
       Royal Blue |    Star Wars 2011 Advent Calendar Poster ||
Star Wars | 0.98
      Medium Blue | Torso Mechanic Blue Overalls, Tools in P ||
City | 1.00
            Black |                            Technic Axle 4 ||
Star Wars Episode 2 | 0.19
      Medium Blue | Torso Mechanic Blue Overalls, Tools in P ||
City | 1.00
```

```
            Black |    Bionicle Mask Kraahkan, Movie Edition ||
Star Wars Episode 2 | 0.29
             Blue |        Legs and Hips [Complete Assembly] ||
City | 1.00
      Trans-Green |                                Plate 1 x 1 ||
Town | 0.92
            Green |          Pine Tree - Large 4 x 4 x 6 2/3 ||
Town | 1.00
       Royal Blue |    Star Wars 2011 Advent Calendar Poster ||
Star Wars | 0.98
            White |                       Road Sign Triangle ||
Town | 0.94
            White |                                Brick 1 x 2 ||
Town | 0.52
             Blue |        Legs and Hips [Complete Assembly] ||
City | 1.00
      Medium Blue | Torso Mechanic Blue Overalls, Tools in P ||
City | 1.00
              Red |                  Bionicle Shoulder Armour ||
Hockey | 0.79
      Medium Blue | Torso Mechanic Blue Overalls, Tools in P ||
City | 1.00
```

The probability for each theme for every tested part (27 themes by 61 tested parts):

```
0.09 |0.01 |0.01 |0.01 |0.01 |0.01 |0.01 |0.01 |0.05 |0.61 |0.02 |
0.00 |0.01 |0.00 |0.00 |0.01 |0.01 |0.00 |0.02 |0.01 |0.00 |0.00 |
0.02 |0.01 |0.05 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.02 |0.00 |0.00 |0.00 |0.01 |0.01 |0.01 |0.00 |0.05 |0.04 |0.00 |
0.00 |0.00 |0.00 |0.01 |0.00 |0.00 |0.01 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.80 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.03 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.96 |0.00 |0.00 |

0.03 |0.00 |0.00 |0.02 |0.00 |0.00 |0.01 |0.00 |0.48 |0.02 |0.01 |
0.00 |0.00 |0.00 |0.00 |0.01 |0.00 |0.00 |0.02 |0.01 |0.00 |0.00 |
0.03 |0.00 |0.32 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.27 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.71 |0.00 |0.00 |

0.06 |0.06 |0.05 |0.00 |0.02 |0.02 |0.02 |0.02 |0.08 |0.06 |0.03 |
0.01 |0.02 |0.02 |0.05 |0.01 |0.04 |0.01 |0.12 |0.11 |0.02 |0.03 |
0.02 |0.01 |0.07 |0.02 |0.03 |
```

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.01 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.16 |0.01 |0.01 |0.04 |0.01 |0.01 |0.01 |0.01 |0.04 |0.07 |0.21 |
0.01 |0.02 |0.01 |0.01 |0.01 |0.01 |0.01 |0.16 |0.06 |0.01 |0.01 |
0.03 |0.02 |0.03 |0.01 |0.01 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.06 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.93 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.16 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.82 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.03 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.96 |0.00 |0.00 |

0.03 |0.00 |0.01 |0.00 |0.02 |0.03 |0.01 |0.01 |0.01 |0.01 |0.01 |
0.50 |0.05 |0.02 |0.09 |0.01 |0.01 |0.01 |0.03 |0.02 |0.01 |0.00 |
0.02 |0.07 |0.01 |0.01 |0.01 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.27 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.71 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.08 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.92 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.01 |0.00 |0.00 |

0.01 |0.01 |0.00 |0.00 |0.01 |0.01 |0.00 |0.00 |0.00 |0.01 |0.00 |
0.03 |0.06 |0.00 |0.76 |0.01 |0.00 |0.01 |0.01 |0.01 |0.00 |0.00 |
0.02 |0.01 |0.00 |0.00 |0.00 |

0.13 |0.01 |0.01 |0.03 |0.01 |0.01 |0.01 |0.01 |0.05 |0.04 |0.18 |
0.01 |0.01 |0.01 |0.01 |0.01 |0.01 |0.01 |0.20 |0.10 |0.01 |0.01 |
0.04 |0.01 |0.05 |0.02 |0.01 |

0.01 |0.00 |0.00 |0.00 |0.01 |0.01 |0.00 |0.00 |0.00 |0.01 |0.00 |
0.79 |0.10 |0.00 |0.02 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.01 |0.01 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.46 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.52 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.01 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.27 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.71 |0.00 |0.00 |

0.07 |0.01 |0.02 |0.00 |0.27 |0.07 |0.01 |0.01 |0.01 |0.02 |0.01 |
0.19 |0.09 |0.01 |0.07 |0.01 |0.01 |0.01 |0.01 |0.01 |0.01 |0.01 |
0.02 |0.01 |0.01 |0.01 |0.01 |

0.18 |0.01 |0.01 |0.03 |0.01 |0.01 |0.01 |0.01 |0.04 |0.02 |0.05 |
0.00 |0.00 |0.01 |0.00 |0.01 |0.01 |0.01 |0.08 |0.04 |0.01 |0.01 |
0.02 |0.01 |0.42 |0.01 |0.01 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.16 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.82 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.01 |0.01 |0.00 |0.00 |0.01 |0.01 |0.00 |0.00 |0.00 |0.01 |0.00 |
0.03 |0.06 |0.00 |0.76 |0.01 |0.00 |0.01 |0.01 |0.01 |0.00 |0.00 |
0.02 |0.01 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.27 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.71 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.01 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.03 |0.00 |0.01 |0.00 |0.02 |0.03 |0.01 |0.01 |0.01 |0.01 |0.01 |
0.50 |0.05 |0.02 |0.09 |0.01 |0.01 |0.01 |0.03 |0.02 |0.01 |0.00 |
0.02 |0.07 |0.01 |0.01 |0.01 |

0.62 |0.01 |0.01 |0.00 |0.02 |0.03 |0.01 |0.01 |0.01 |0.02 |0.01 |
0.02 |0.03 |0.01 |0.05 |0.01 |0.01 |0.00 |0.05 |0.03 |0.01 |0.01 |
0.02 |0.01 |0.00 |0.01 |0.01 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.18 |0.01 |0.01 |0.03 |0.01 |0.01 |0.01 |0.01 |0.04 |0.02 |0.05 |
0.00 |0.00 |0.01 |0.00 |0.01 |0.01 |0.01 |0.08 |0.04 |0.01 |0.01 |
0.02 |0.01 |0.42 |0.01 |0.01 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.01 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.01 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.15 |0.01 |0.01 |0.03 |0.01 |0.01 |0.01 |0.01 |0.14 |0.03 |0.07 |
0.00 |0.01 |0.01 |0.01 |0.01 |0.01 |0.01 |0.19 |0.04 |0.01 |0.01 |
0.08 |0.01 |0.11 |0.01 |0.01 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.07 |0.01 |0.01 |0.02 |0.01 |0.01 |0.01 |0.01 |0.15 |0.03 |0.13 |
0.00 |0.00 |0.01 |0.01 |0.01 |0.01 |0.01 |0.29 |0.05 |0.01 |0.01 |
0.05 |0.01 |0.06 |0.01 |0.01 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.08 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.92 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |1.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.98 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.01 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.05 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.94 |0.00 |0.00 |

0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.46 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.52 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

0.02 |0.01 |0.01 |0.00 |0.01 |0.01 |0.01 |0.01 |0.01 |0.79 |0.01 |
0.01 |0.01 |0.00 |0.01 |0.00 |0.01 |0.00 |0.01 |0.01 |0.01 |0.01 |
0.00 |0.01 |0.02 |0.00 |0.01 |

0.00 |0.00 |0.00 |1.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |0.00 |
0.00 |0.00 |0.00 |0.00 |0.00 |

Please adjust window width to fit formatting of 100 characers.
===== END OF S14139866 LEGO BRICKS AI =====