

Project 4

Board Games & Machine Learning

Priscila Briggs, Frances Jay,
John Kiersznowski,
Sea Gun Lee, & Zac Link



Board Games

- Board games have been played in nearly all societies' cultures throughout human history. There are many different styles and genres. **Games can be based on strategy, chance, or a combination of the two** and usually have a goal that players try to achieve before their opponent(s).
- Even with video games, streaming services, and other forms of electronic entertainment, many people still turn to board games for a budget-friendly way to spend time with family and friends. **The pandemic helped to increase the popularity of board games**, as many people needed something to do while they were stuck inside; **the board game market grew by 20% in 2020**, and revenue from board games sales is expected to hit the \$12 billion mark in 2023.

The 10 Top-Selling Board Games

1. **Chess**: Total number of units sold is unknown; > 3 million units sold yearly in the U.S. alone
2. **Checkers**: 50 billion units sold since introduction
3. **Monopoly**: More than 275 million units sold
4. **Scrabble**: More than 150 million units sold
5. **Clue**: More than 150 million units sold
6. **Battleship**: More than 100 million units sold
7. **Trivial Pursuit**: More than 100 million units sold
8. **Backgammon**: More than 88 million units of the modern version sold
9. **Candy Land**: More than 50 million units sold
10. **Rummikub**: More than 50 million units sold

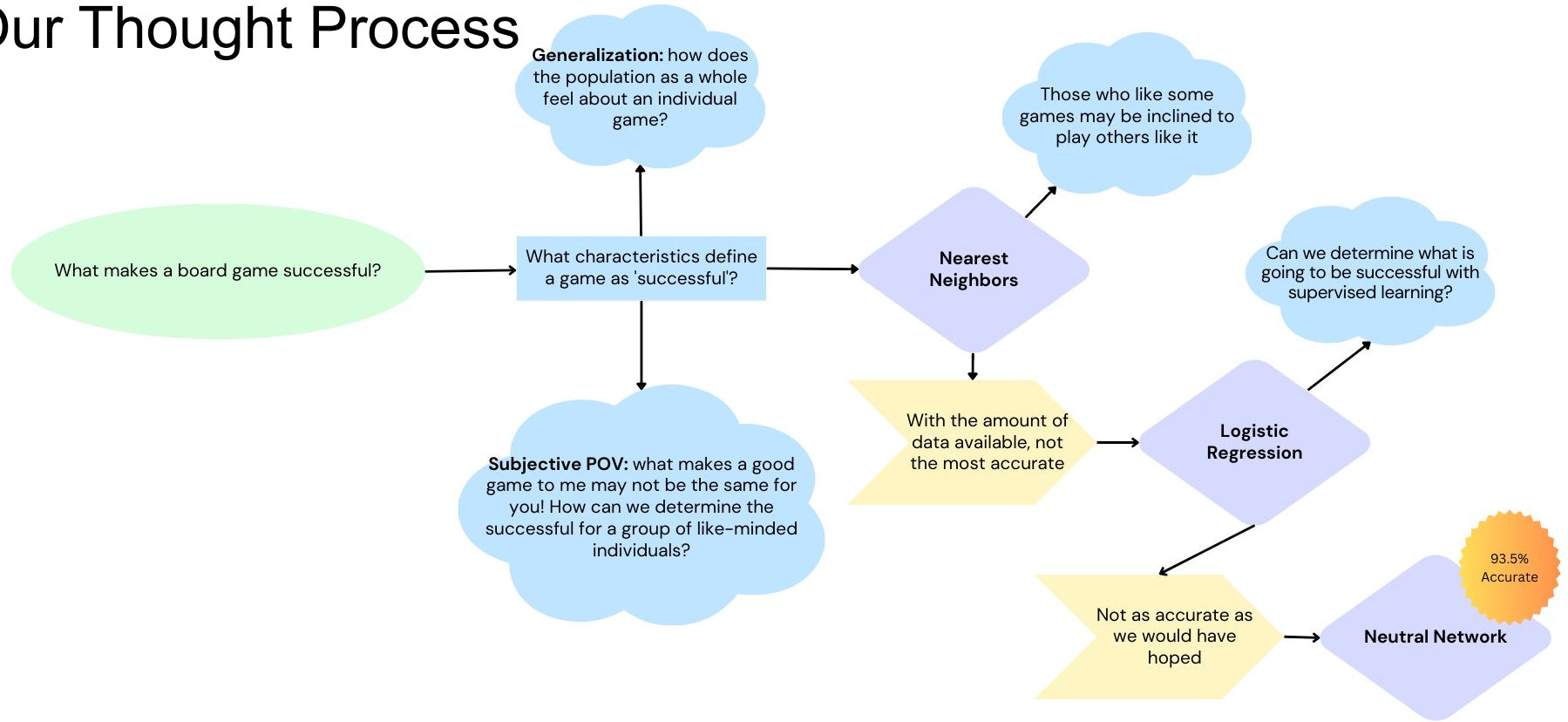


Challenge:

As a consulting firm, could we generate a model/analysis for a global board game creator/distributor that could encompass the attributes of a board game, and accurately depict whether that game would be successful?



Our Thought Process



Data Collection

- Board Game Database from BoardGameGeek
- <https://www.kaggle.com/datasets/threnjen/board-games-database-from-boardgamegeek?select=themes.csv>
- BGG provided several CSV files containing information on each game, the themes, subcategories, and mechanics available (and had already split out the categorical data into dummy variables)

| BGGId | Name |
|-------|--------------------|
| 1 | Die Macher |
| 2 | Dragonmaster |
| 3 | Samurai |
| 4 | Tal der Könige |
| 5 | Acquire |
| 6 | Mare Mediterraneum |
| 7 | Cathedral |
| 8 | Lords of Creation |
| 9 | El Caballero |
| 10 | Elfenland |
| 11 | Bohnanza |
| 12 | Ra |
| 13 | Catan |
| 14 | Basari |
| 15 | Cosmic Encounter |
| 16 | MarraCash |
| 17 | Button Men |
| 18 | RoboRally |
| 19 | Wacky Wacky West |
| 20 | Full Metal Planète |

Data Preparation

- CSV read into pandas DataFrame
- Standard scaler was used to scale all numerical data
- Cleaned up NaN values
- Removed columns that did not provide value (image path, NumComments, etc)

YearPublished GameWeight MinPlayers MaxPlayers ComAgeRec LanguageEase NumWeightVotes MfgPlaytime ComMinPlaytime ComMaxPlaytime

BGGId

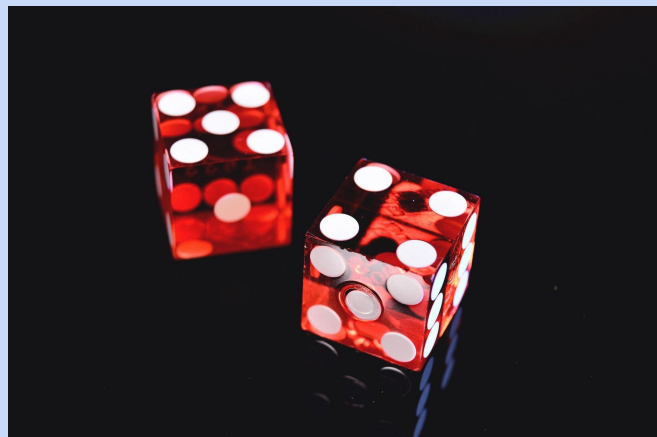
| | | | | | | | | | | |
|---|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 1 | 0.002377 | 2.754499 | 1.432245 | -0.047146 | 1.341048 | -0.700713 | 3.458048 | 0.282239 | 0.397204 | 0.282239 |
| 2 | -0.021154 | -0.022535 | 1.432245 | -0.113749 | 0.634536 | -0.586369 | 0.021967 | -0.114253 | -0.075869 | -0.114253 |
| 3 | 0.058853 | 0.593393 | -0.010595 | -0.113749 | -0.169188 | -0.702480 | 6.811507 | -0.057611 | -0.075869 | -0.057611 |
| 4 | 0.030615 | 0.806359 | -0.010595 | -0.113749 | 0.933062 | 0.436308 | -0.094675 | -0.057611 | -0.008287 | -0.057611 |
| 5 | -0.101161 | 0.613653 | -0.010595 | 0.019457 | 0.458483 | -0.612481 | 7.564820 | -0.000970 | 0.059295 | -0.000970 |

5 rows × 422 columns

Nearest Neighbors - Unsupervised Learning

Overview - Unsupervised machine learning

- 20,000+ board games
- Learning method: nearest neighbors
- Objective: recommendation of board games with similarities
- Relevant features:
 - Name, description, game weight (difficulty), avg. rate etc
 - Categories:
 - Thematic
 - Strategy
 - War
 - Family
 - Card Games (CGS)
 - Abstract
 - Party
 - Children



Feature Engineering / Model Implementation

Nearest neighbors

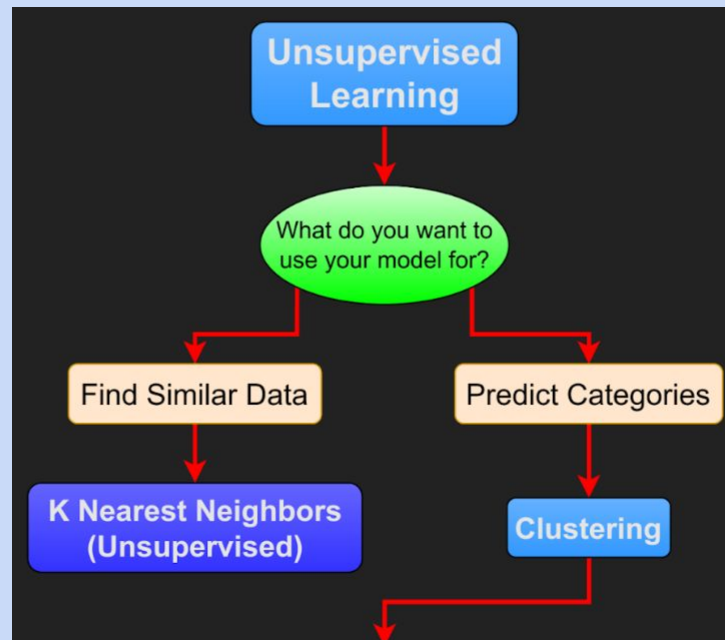
- Features with info related to players or images were removed, as well as columns with several null values or complete lack of data
- Model trained on scaled data, default algorithm = 'auto', resulting on 5 recommendations

```
1 from sklearn.neighbors import NearestNeighbors
2 neigh = NearestNeighbors(n_neighbors=6)
3 neigh.fit(nn_df_transformed.iloc[:,25:39])
```

▼ NearestNeighbors

NearestNeighbors(n_neighbors=6)

- Considerations:
 - Because the query set matches the training set, the nearest neighbor of each point is the point itself, at a distance of zero.
 - Regarding the Nearest Neighbors algorithms, if two neighbors $k+1$ and k have identical distances but different labels, the result will depend on the ordering of the training data.



Source: PACHECO, AHMAD, PRUITT. *Machine Learning Road Map: A guide for non-PHDs*.

Example Run of Nearest Neighbors Logic

Nearest Neighbor Results

Monopoly

| | BGGId | Name | Description | YearPublished | GameWeight | AvgRating | Cat:Thematic | Cat:Strategy | Cat:War | Cat:Family | Cat:CGS | Cat:Abstract | Cat:Party | Cat:Childrens |
|------|-------|--|---|---------------|------------|-----------|--------------|--------------|---------|------------|---------|--------------|-----------|---------------|
| 567 | 684 | Monopoly: The Card Game | rummy monopoly fan property board card collect form complete set house hotel token card multiply value set mr monopoly card provide quick point help fill card hand player game instead discard pile player trade pile player draw deck exchange trade pile player like stack money include unnecessary cash score spend exchange | 2000 | 1.4681 | 5.53780 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 3758 | 6610 | Spinergy | boxspinergy give chance writer actor singer poet date magnet standup comic give random word describe life save platoon marinescompose shocking tabloid headline deliver ultimate pickup line create blockbuster movie haiku riddle joke tongue twister song foreign accent different time play | 2000 | 1.4167 | 5.68641 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5862 | 14379 | Harry Potter and the Sorcerer's Stone Mystery at Hogwarts Game | cluestyle game come game board fluffy folder hold solution special dice wizard hat pawn ghost pawn hogwart event card character card magic card room card summary card check list paddle object game deduce student cast forbid spell room hogwart school player think heshe know solution travel difficult reach floor fluffy guard answerwhile mechanic play familiar player classic clue new element game require slightly different approach play | 2000 | 1.5161 | 5.54027 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Chess

| | BGGId | Name | Description | YearPublished | GameWeight | AvgRating | Cat:Thematic | Cat:Strategy | Cat:War | Cat:Family | Cat:CGS | Cat:Abstract | Cat:Party | Cat:Childrens |
|------|-------|---------------|--|---------------|------------|-----------|--------------|--------------|---------|------------|---------|--------------|-----------|---------------|
| 31 | 32 | Buffalo Chess | buffalo chess aka bison player represent indian try village overrun buffalo player move indian chief dog player take charge herd rampage buffalo represent wooden piece style play piece differently | 1975 | 1.5417 | 6.06147 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5120 | 10845 | Napoleon | pelikan classic buchkassette series napoleon lightlytheme player wargame checkerslike movement rule | 1975 | 1.6667 | 5.98438 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1743 | 2427 | Skirmish | american heritage command decision strategy game highly prize collectorsthis american war independence game british army unit warship troop carrier reinforcement board americans unit spread board include special quotcontinental armyquot unit warship troop carrier reinforcement boardthe americans use dice movement unit british dice player roll dice shaker peek british player designate unit hell american reveal throw move finally british reveal throw movesa single unit move die grid square diagonal allow unit end spot stack form army split army limit unit naval move harbour harbour troop carrier warship attempt intercept thema skirmish occur oppose unit adjacent card draw resolve skirmish result unit loss major battle trigger army move space major battle normally occur army land enemy army exact count numerical superiority add strong side throw adjust throw difference indicate unit lose lose loss limit number unit win armyonce british lose army reinforcement americans ve win major battle reinforcement play resolve separately normal play dice side player try bring troop carrier whilst attempt intercept warship opponent warship leave reinforcement automatically succeed troop carrier make harbor reinforcement land trigger major battle harbor enemyheld troop carrier remain normal warship henceforththe british win eliminate continental army whilst american win eliminate british army game drag skirmish major battle move weak surrenders the asymmetrical mechanic work surprisingly british lack mobility try force major battle americans whilst skirmish possible muster large army win major battle reinforcement | 1975 | 1.7500 | 6.21154 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

Model Evaluation / Adjustment

- Scaled data (normalized the data)
- Attempts with different number of related features

Statistical Accuracy

- Gross calculation (manual) of matching categories: %

Recommendation

- Use in larger datasets

Logistic Regression Supervised Learning

Segway into Supervised ML (Logistic Regression)

- Target variable: Average rating
- Feature variables:
 - Name, Year Published, Min / Max Players, Average Rate, categories (adventure, fantasy, etc)



Feature Engineering / Model Implementation

- Board games labeled as good (1) or bad (0) based on mean avg rating (6.42) & median avg rating (6.45)
 - `DataFrame["AvgRating"].apply(lambda x: 1 if x > 6.5 else 0)`
- Data was split, trained, and tested based on target (AvgRating)



Logistic Regression Performance

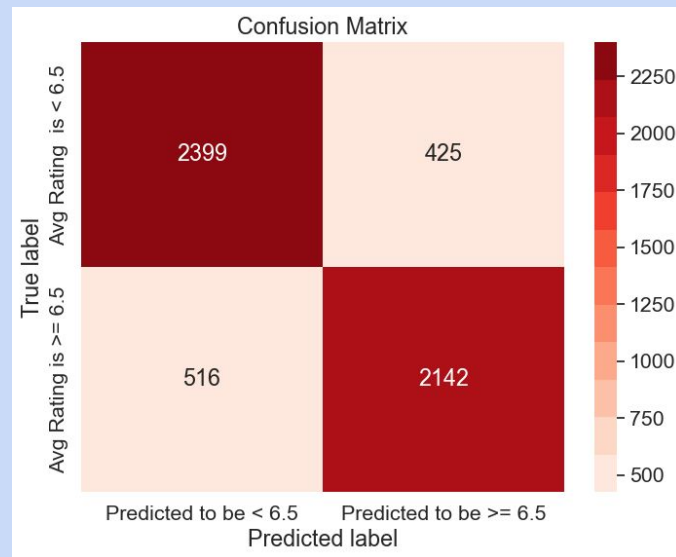
- We identified a few tweaks that could be made to our initial model to make it more accurate:
 - Scaled data (normalized the data)
 - Changed the sklearn logistic regression default solver (lbfgs) to sag
 - Increased max iterations from default, 1000 to 5000
- In doing so, we brought the accuracy of the logistic regression model up from 66% to 83%

Logistic Regression Accuracy

Classification Report



Confusion Matrix



Neural Network

- **Accuracy Score - 83%**

Using our knowledge of Deep Learning,
can we improve our Accuracy Score?

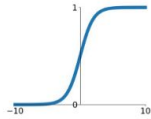


Neural Network - Deep Learning

Activation Functions

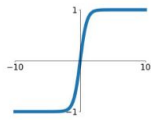
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



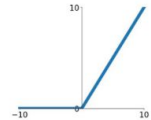
tanh

$$\tanh(x)$$



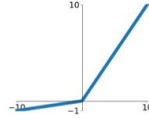
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

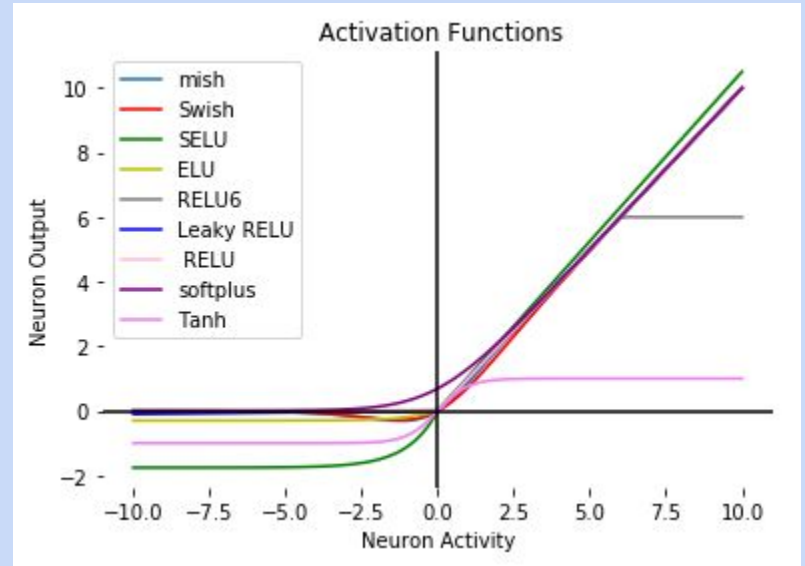
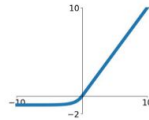


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Fine Tuning the Model

```
In [6]: 1 #lets try a different model builder
2 import tensorflow as tf
3 from tensorflow import keras
4
5 #!pip install -q -U keras-tuner
6
7 import keras_tuner as kt
8
9
10 def model_builder(hp):
11     model = keras.Sequential()
12     model.add(keras.layers.Flatten(input_dim=33))
13
14     # Tune the number of units in the first Dense layer
15     # Choose an optimal value between 32-512
16     hp_units = hp.Int('units', min_value=2, max_value=50, step=2)
17     model.add(keras.layers.Dense(units=hp_units, activation='selu'))
18     model.add(keras.layers.Dense(10))
19
20     # Tune the learning rate for the optimizer
21     # Choose an optimal value from 0.01, 0.001, or 0.0001
22     hp_learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-1])
23
24     model.compile(optimizer=keras.optimizers.Adam(learning_rate=hp_learning_rate),
25                   loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
26                   metrics=['accuracy'])
27
28     return model
```

Fine Tuning the Model

```
In [8]: 1 stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
        2
        3 tuner.search(X_train, y_train, epochs=10, validation_split=0.2, callbacks=[stop_early])
        4
        5 # Get the optimal hyperparameters
        6 best_hps=tuner.get_best_hyperparameters(num_trials=1)[0]
        7
        8 print(f"""
        9 The hyperparameter search is complete. The optimal number of units in the first densely-connected
       10 layer is {best_hps.get('units')} and the optimal learning rate for the optimizer
       11 is {best_hps.get('learning_rate')}.
       12 """)
```

```
Trial 90 Complete [00h 00m 34s]
val_accuracy: 0.9087868928909302
```

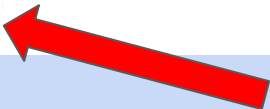
```
Best val_accuracy So Far: 0.9252052307128906
Total elapsed time: 00h 11m 12s
```

```
The hyperparameter search is complete. The optimal number of units in the first densely-connected
layer is 18 and the optimal learning rate for the optimizer
is 0.01.
```

Outcome

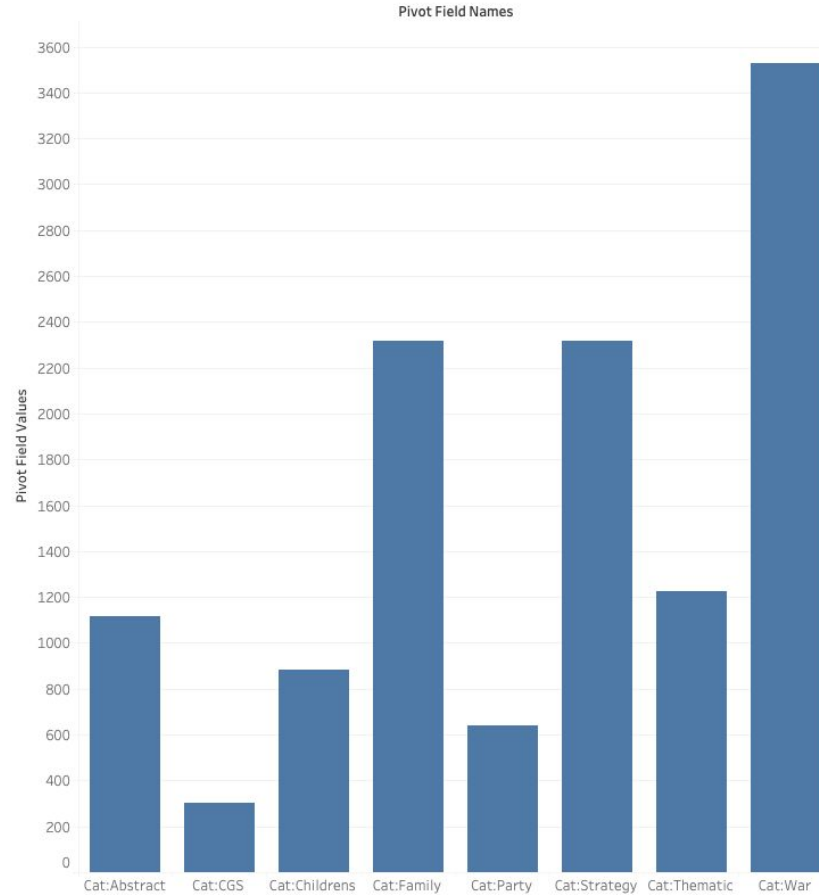
```
1 # Define the deep learning model
2 nn_model = tf.keras.models.Sequential()
3 nn_model.add(tf.keras.layers.Dense(units=16, activation="ELU", input_dim=33))
4 nn_model.add(tf.keras.layers.Dense(units=16, activation="ELU"))
5 nn_model.add(tf.keras.layers.Dense(units=16, activation="ELU"))
6 nn_model.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))
7
8 # Compile the Sequential model together and customize metrics
9 nn_model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
10
11 # Train the model
12 fit_model = nn_model.fit(X_train, y_train, epochs=50)
13
14 # Evaluate the model using the test data
15 model_loss, model_accuracy = nn_model.evaluate(X_test, y_test, verbose=2)
16 print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

172/172 - 0s - loss: 0.2292 - accuracy: 0.9305 - 297ms/epoch - 2ms/step
Loss: 0.22916701436042786, Accuracy: 0.930499792098999



Neural Network Category Investigation

Categories Count

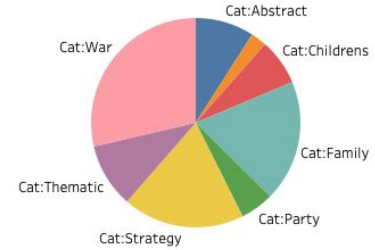


Category Proportion Percentage

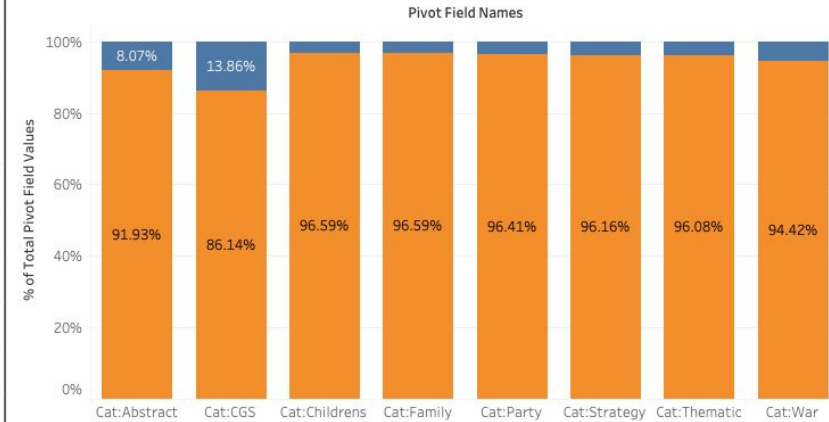
Pivot Field Names

| | |
|---------------|--------|
| Cat:War | 28.63% |
| Cat:Strategy | 18.81% |
| Cat:Family | 18.79% |
| Cat:Thematic | 9.93% |
| Cat:Abstract | 9.04% |
| Cat:Childrens | 7.15% |
| Cat:Party | 5.19% |
| Cat:CGS | 2.46% |

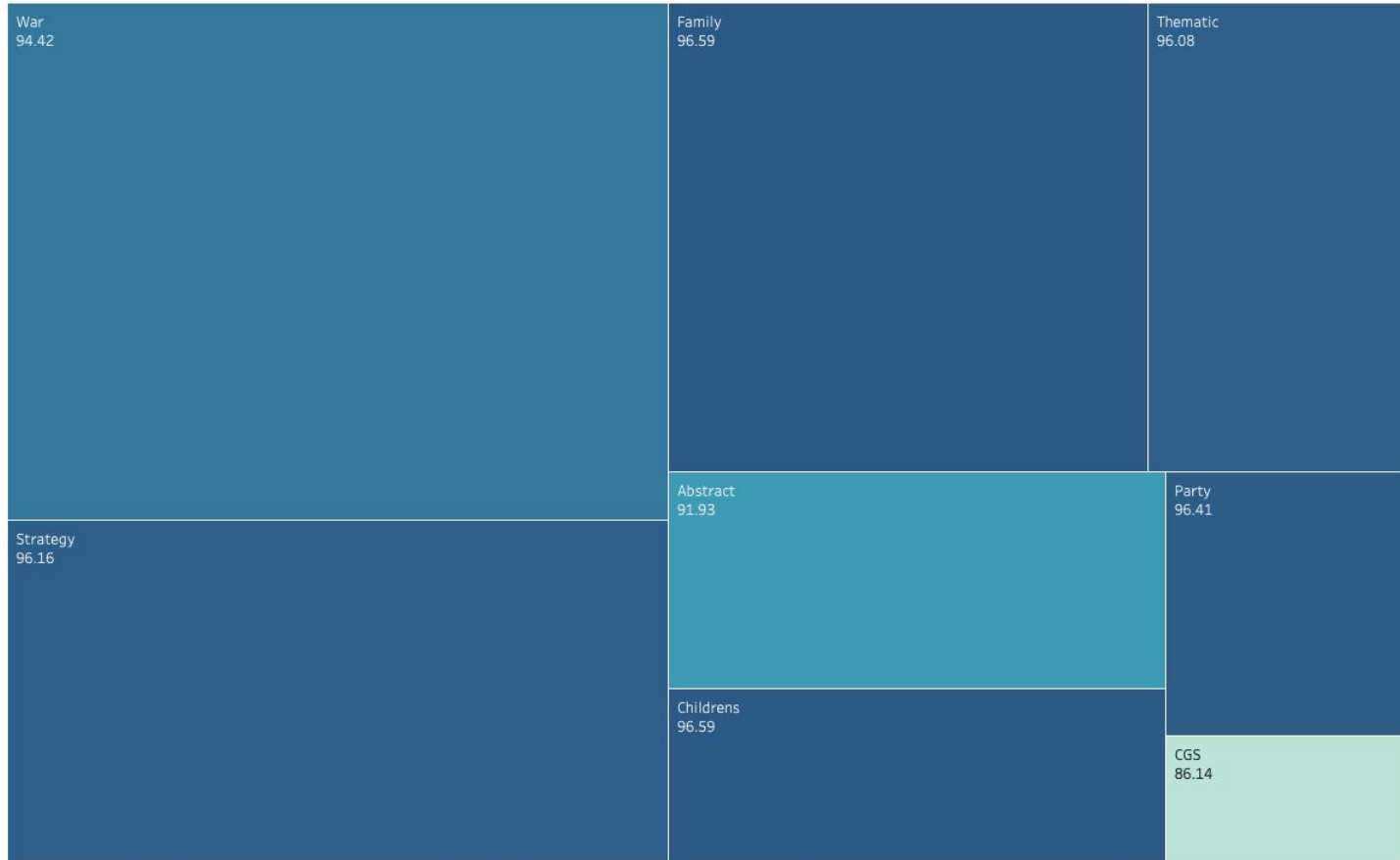
Category Proportion Pie Chart



Category Accuracy



Accuracy by Category with Proportion Visual



SUM(TRUE Prediction ...)

86.14 96.59