Prediction of Magic: The Gathering cards

Classification and Regression methods

Group 4:
August Heegaard, Jacob Hallberg, Laust Rask

UNIVERSITY OF COPENHAGEN

Outline

- Motivation
- Data overview
- Classification using images
- Human Benchmarks
- Classification using text
- Regression using text
- Conclusion & Outlook

Motivation

- Magic: The Gathering is widely known!
- Lots of data
- Accessible data
- Both meta and in-game data
- Lot of color scheme and interesting artwork
- Human interpretable data
- Different kinds of models can be utilized

Data Presentation

Card Name

Type Line

Keywords

Flavor Text

Mana Cost

Card Image

Rarity

Oracle Text

Power/ Toughness

-5 Different colors
-Each with pervasive features
-A card can be between 0 and 5 colors

Data Presentation

-Total of 14178 cards
-78 different sets
-Cards from between 2004 and 2024

Data Acquisition

Data acquired using Scryfall API – a python plugin

114 columns

cmc	object	id	oracle_id	mtgo_foil_id	tcgplayer_id	name	lang	…	waterm	loyalty

11 columns

Name	mana_cost	cmc	type_line	oracle_text	Power	Toughness	Keywords	Rarity	Prices	RGB
Knight Exemplar	{1}{W}{W}	3	Creature – Human Knight	Other Knight creatures…	2	2	[First strike]	Rare	0.05	[18,17,…], [95,25,…], [255,243,…]

Features and Labels

One-Hot → ['W','U','B','R','G'] → [1, 0 , 0, 0, 0]

$2^5 = 32$

"Classic" → [C, W, U, B, R, G, WU, WB ... WUBRG]

32

Features → (256,256,3) → Input

One-Hot CNN – Color Build

- Input: **Image (256,256,3) shape**
- Structure:

CONV → POOL → NORM → (CONV→POOL)x2 → DENSE → SIGMOID

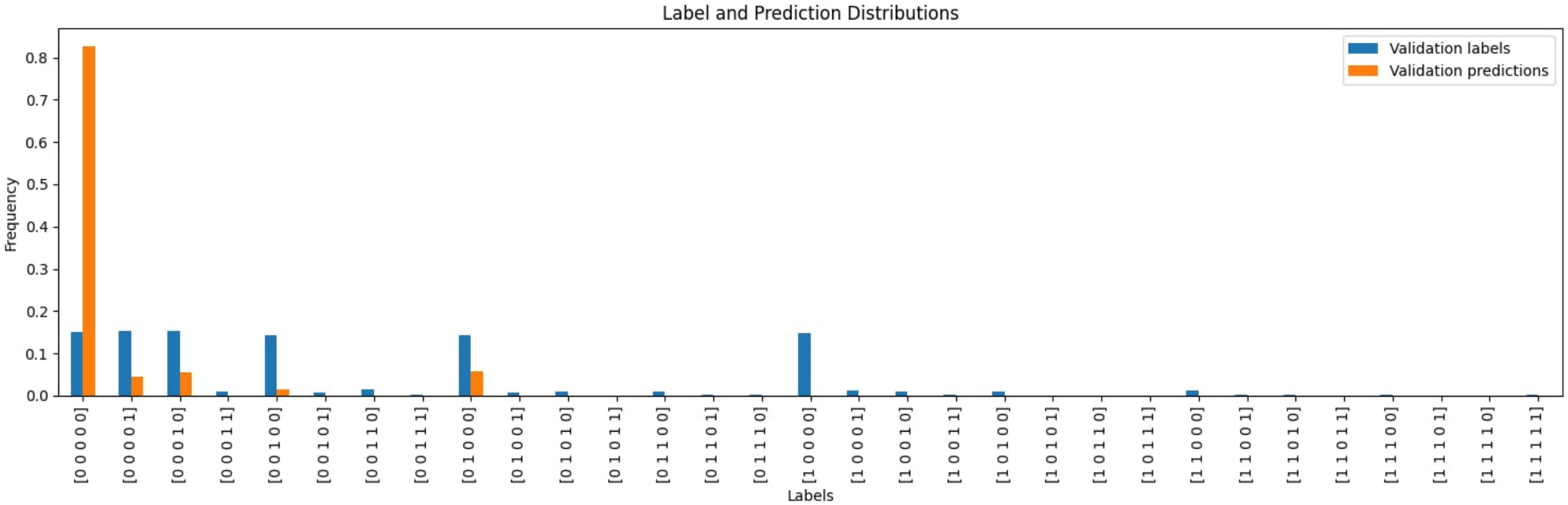
- Optimizer: **ADAM**
- Kernel Size: **(3, 3)**, Pool Size: **(2, 2)**
- Loss: **Weighted Binary Cross Entropy**

- Problem: Lots of 0's

One-Hot CNN - Results

Test Accuracy: 0.16
Train Accuracy: 0.18

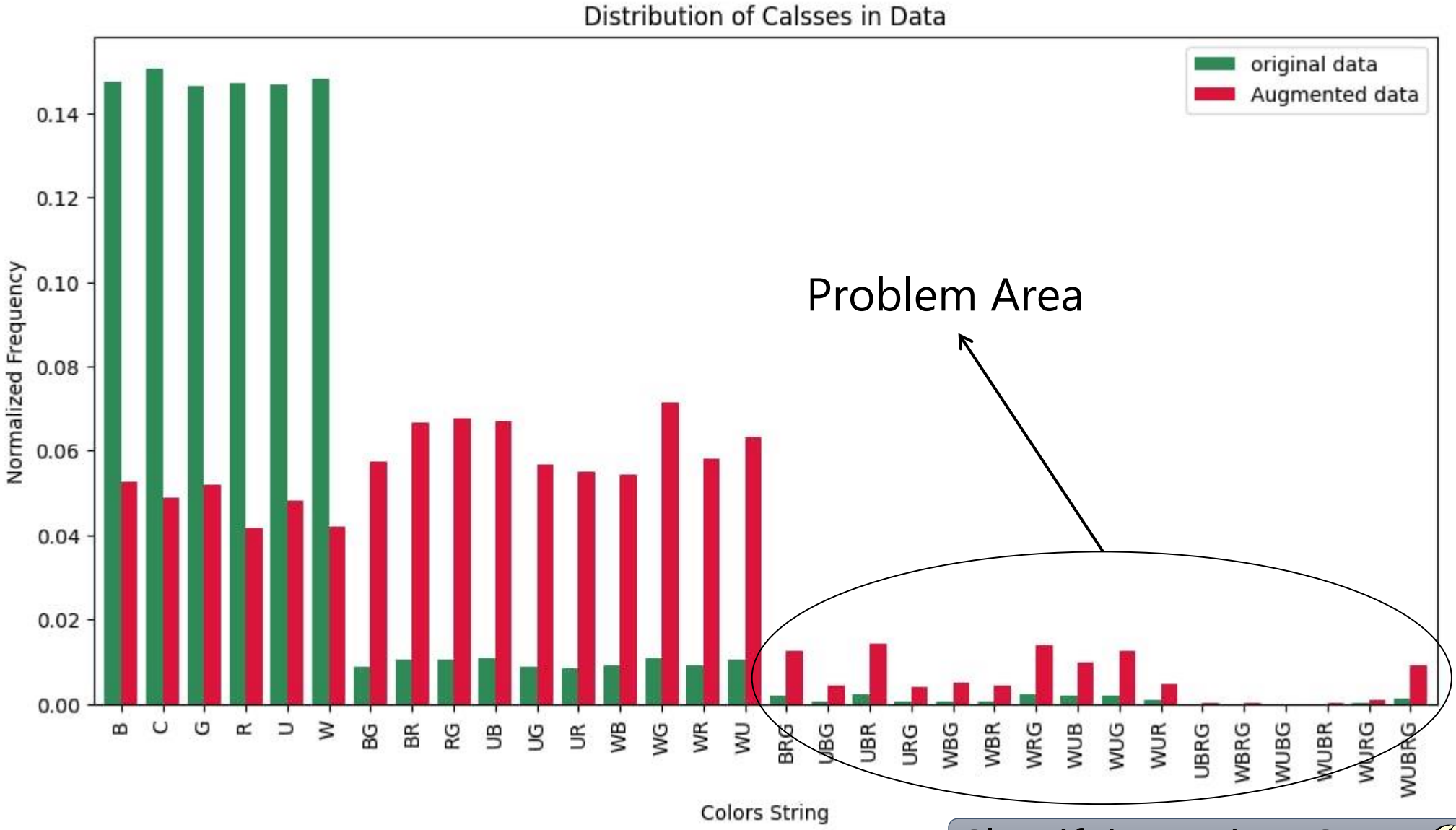
One-Hot: [1,0,0,0,0] → 95% 0's



“Classic” CNN – Data Augmentation

Data Augmentation:

Data Augmentation - Distribution



"Classic" CNN – Color Build

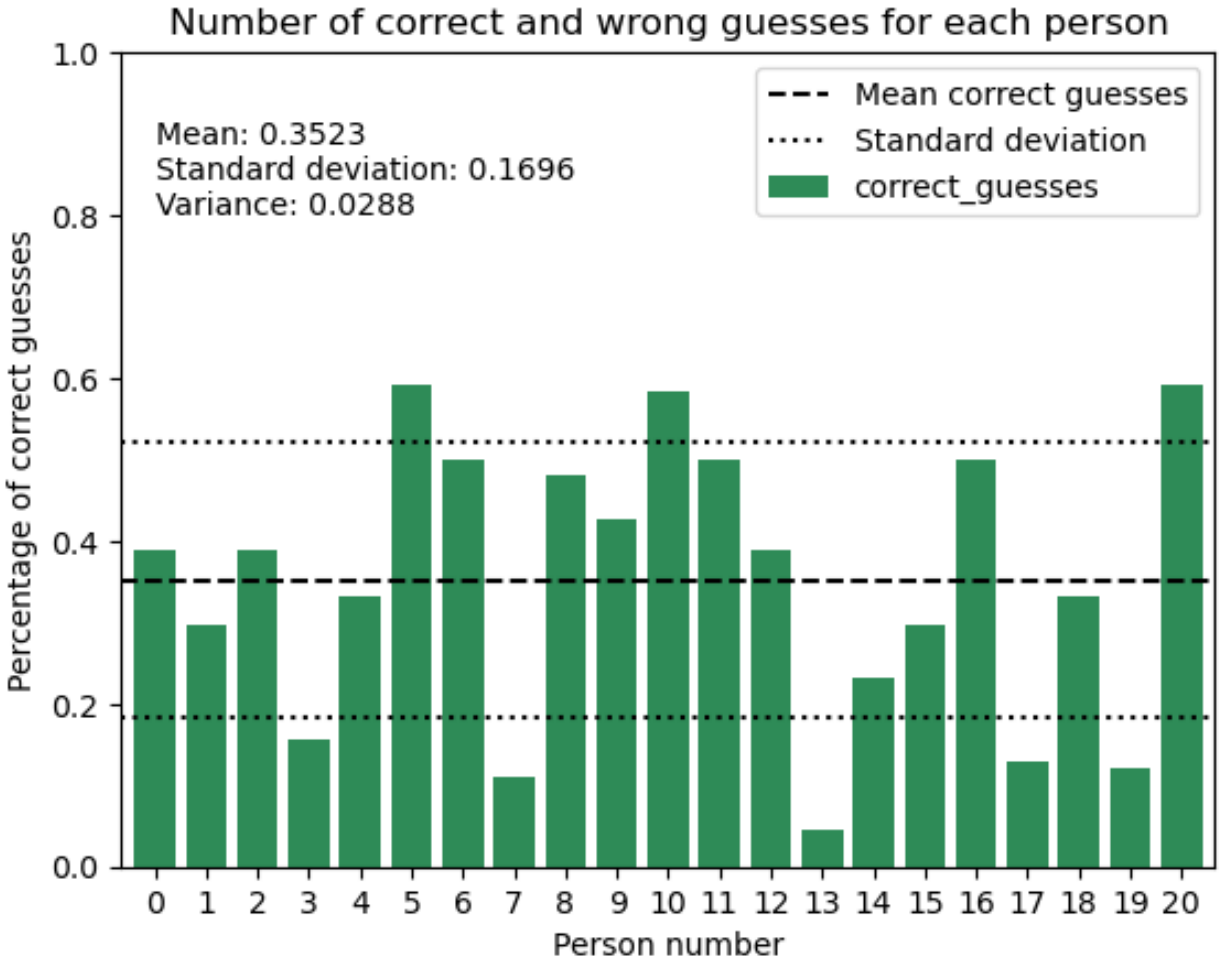
- Input: **Image (256,256,3) shape**
- Structure: (**CONV → POOL → NORM)x3 → DENSE → SOFTMAX**
- Optimizer: **ADAM**
- Loss: **Categorical Cross Entropy**
- Kernel Size: **(3, 3)**, Pool Size: **(2, 2)**
- Bayesian Optimization:

Test Accuracy: 0.45
Train Accuracy: 0.93

```
hyperparameter_space = {
    'dropout_rate': (0.1, 0.5),
    'l2_reg'      : (1e-6, 1e-4)
    'conv1'       : (16, 64),
    'conv2'       : (32, 128),
    'conv3'       : (32, 128),
    'dense_size'  : (32, 128),
    'lr'          : (0.001,0.01)
```

```
best_hyperparameter = {
    'dropout_rate': 0.266,
    'l2_reg'      : 5.73e-5,
    'conv1'       : 28,
    'conv2'       : 106,
    'conv3'       : 75,
    'dense_size'  : 82,
    'lr'          : 0.00117}
```

CNN vs. Human benchmark - Color



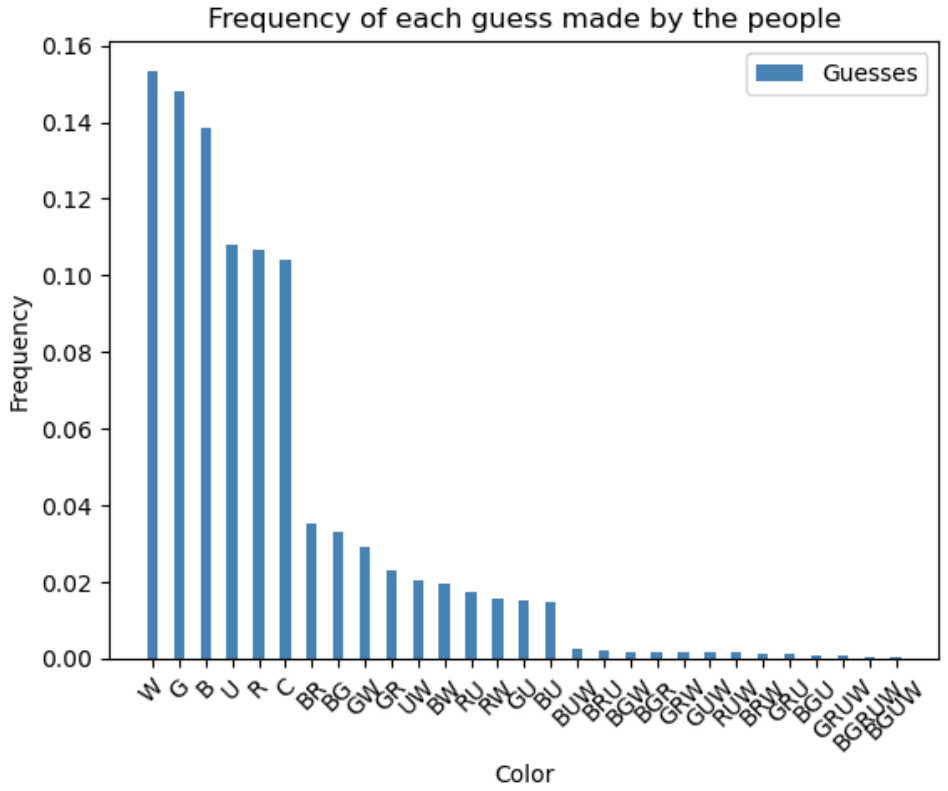
- Questionnaire
- Train set (48 pics)
- Test set (108 pics)
- Only Images
- Same set

CNN vs. Human benchmark

Hardest: Pacifism (0%) **Easiest:** Brindle Boar (81%)

Evaluator	Color Guess
True	W
Human	B
One-Hot CNN	C
“Classic” CNN	W

Evaluator	Color Guess
True	G
Human	G
One-Hot CNN	C
“Classic” CNN	G

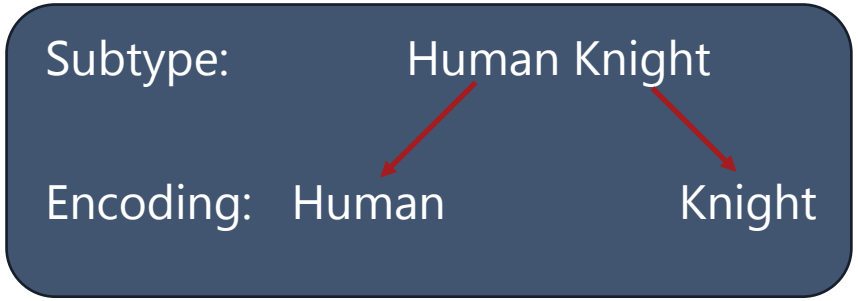


Feature engineering – From Text to Numerical

Name	mana_cost	cmc	type_line	oracle_text	Power	Toughness	Keywords	Rarity	Prices
Knight Exemplar	{1}{W}{W}	3	Creature – Human Knight	Other Knight creatures…	2	2	[First strike]	Rare	0.05

One-hot encoding of:

- Mana Cost
- Type Line
  - Cardtypes (35)
  - Subtypes (298)
- Power & Toughness (Imputed: -1)
- Keywords (213)
- Rarity (4)



Normalization:

- Mana Cost
- Power & Toughness
- Price (USD)

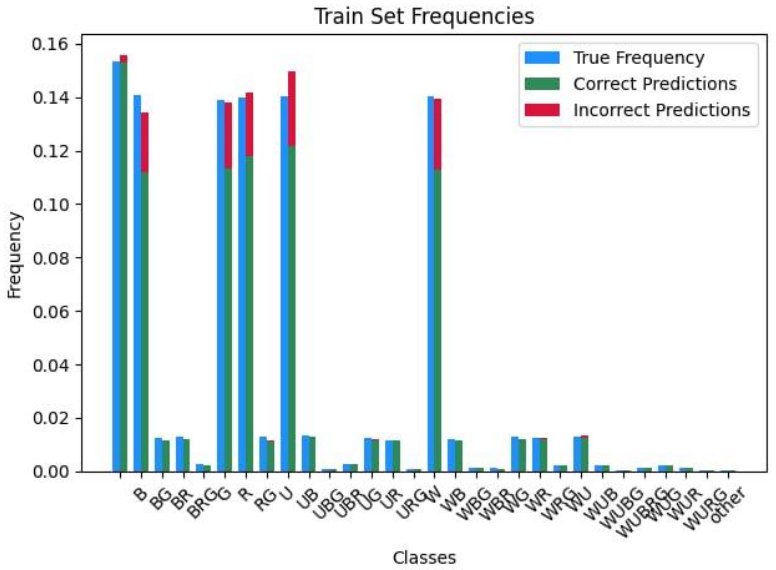
Vectorization of Oracle Text:

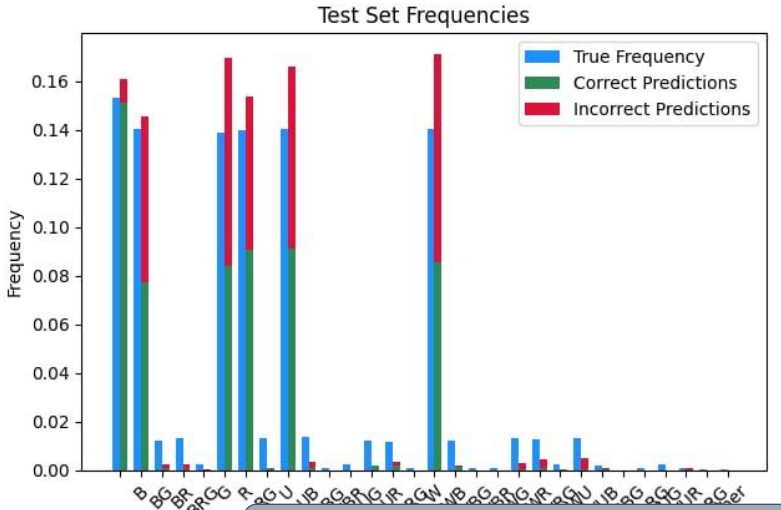
- Pretrained Fast Text
- Embedding of words/sentences
- Own name replaced by “CARDNAME”
- Vector size: 300

XGBoost – Color Classifcation

```
XGBoost_Parameters = {
        'n_estimators'     : 800,
        'max_depth'        : 3,
        'lr'               : 0.3,
        'subsample'        : 0.8,
        'colsample_bytree' : 0.8,
        'reg_alpha'        : 0.5,
        'reg_lambda'       : 1.0}
```

Loss function: Multiclass LogLoss





Model Evaluation

- Test accuracy: 59.12 %
- Train accuracy: 86.90 %

XGBoost – Regression & Classifier on Attributes

Card Prizes:

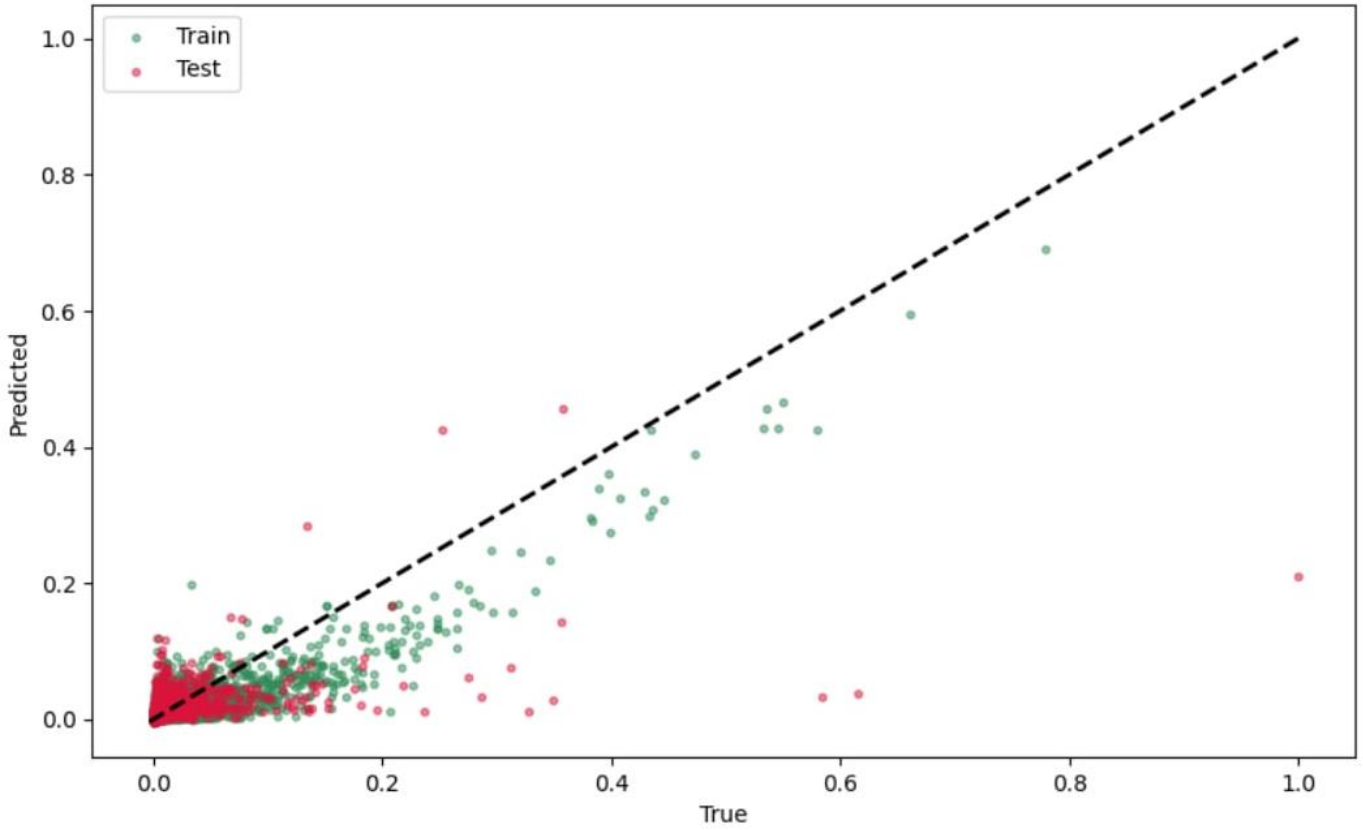
- Test R2: 0.2 – Outliers Removed: 0.3
- Train R2: 0.82 – Outliers Removed: 0.89

Rarity Classification:

- Test Accuracy: 0.699
- Train Accuracy: 0.934

Mana Cost Regression:

- Test Accuracy: 0.539
- Train Accuracy: 0.979



Conclusion and Outlook

Human vs ML

The average accuracy on the test set was 35% -
16% standard deviation

Humans had some consistencies in which cards were easy/difficult to predict

Our model was better than the average human, but a "Fantasy"-lover is better!

Encoding Method

One-hot encoding was *not* viable due to "bit imbalance" – to a point where over sampling was not viable

32 outputs was majorly **imbalanced**, to a point where over sampling was not enough

Accuracy of model

Color predictions:

CNN : 45%

XGBClassifier : 58%

CMC:

XGBRegressor : 43%

Rarity:

CNN: 37% -worse than mode

XGBClassifier : 70%

Price USD:

XGBRegressor: R2 = 0.3

Improvements

Combine the **CNN** and **Text** classifier –
Trying this improved from 50% to 55% accuracy

Auto-encoding for more over sampling of underrepresented classes

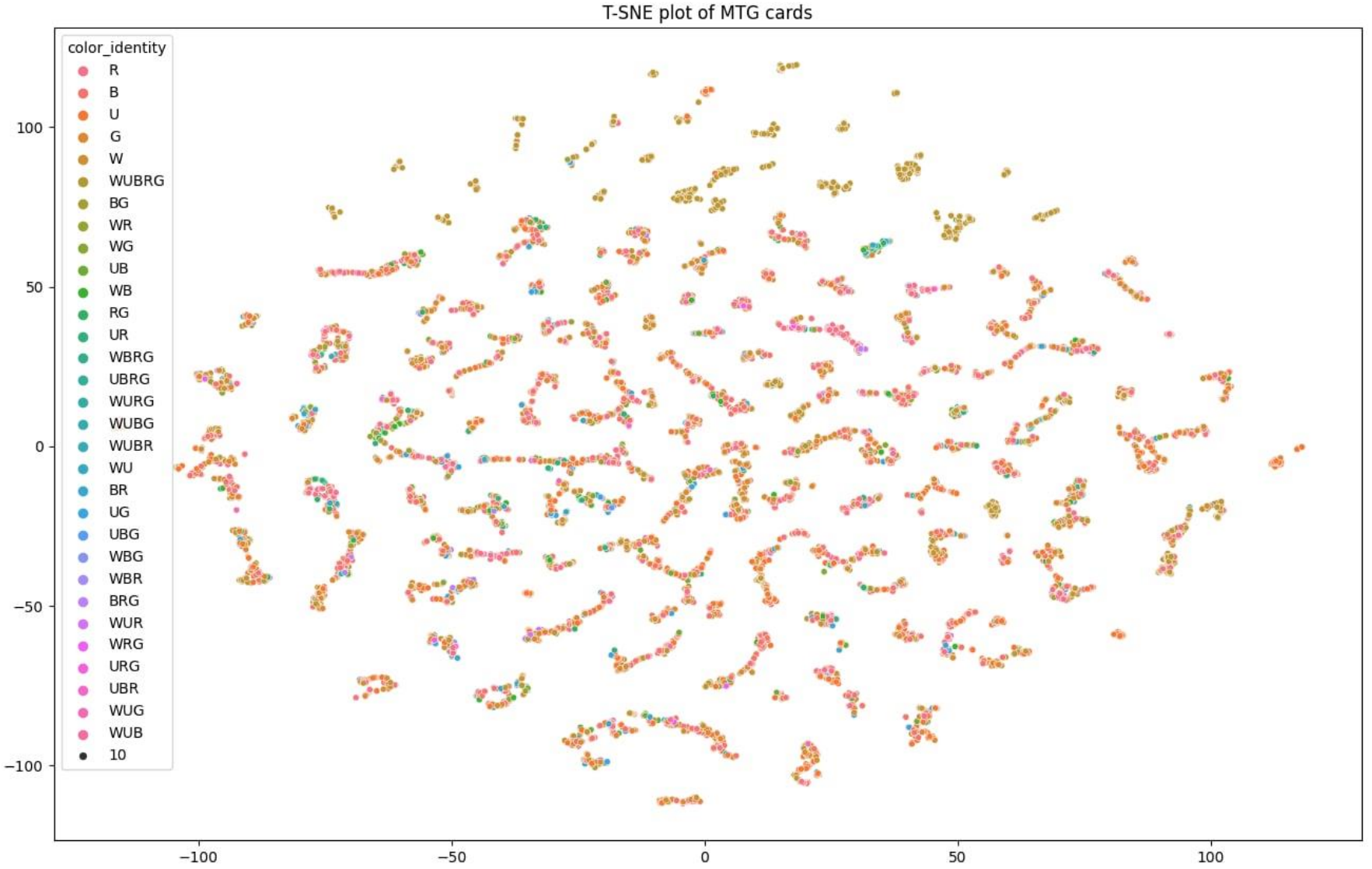
MTG **specific** oracle **encoder**

Potentially **generating new cards**, based on cost and/or other requirements

Questions & comments?

APPENDIX

Unsupervised t-SNE of text data



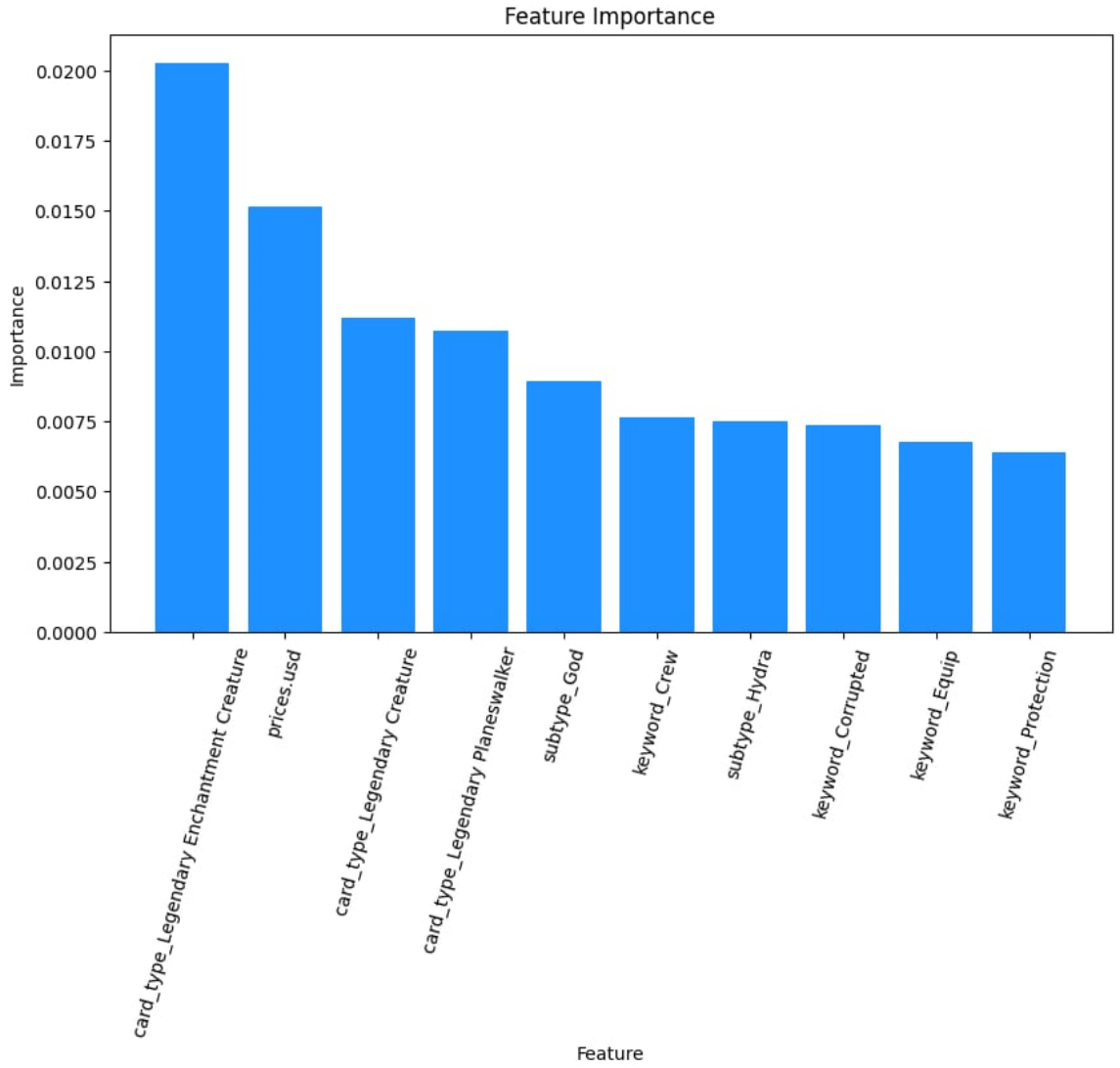
XGBoost – Rarity Build

```
XGBoost_Parameters = {
        'n_estimators'     : 600,
        'max_depth'        : 5  ,
        'lr'               : 0.1,
        'subsample'        : 0.8,
        'colsample_bytree' : 0.8,
        'reg_alpha'        : 0.3,
        'reg_lambda'       : 0.4}
```

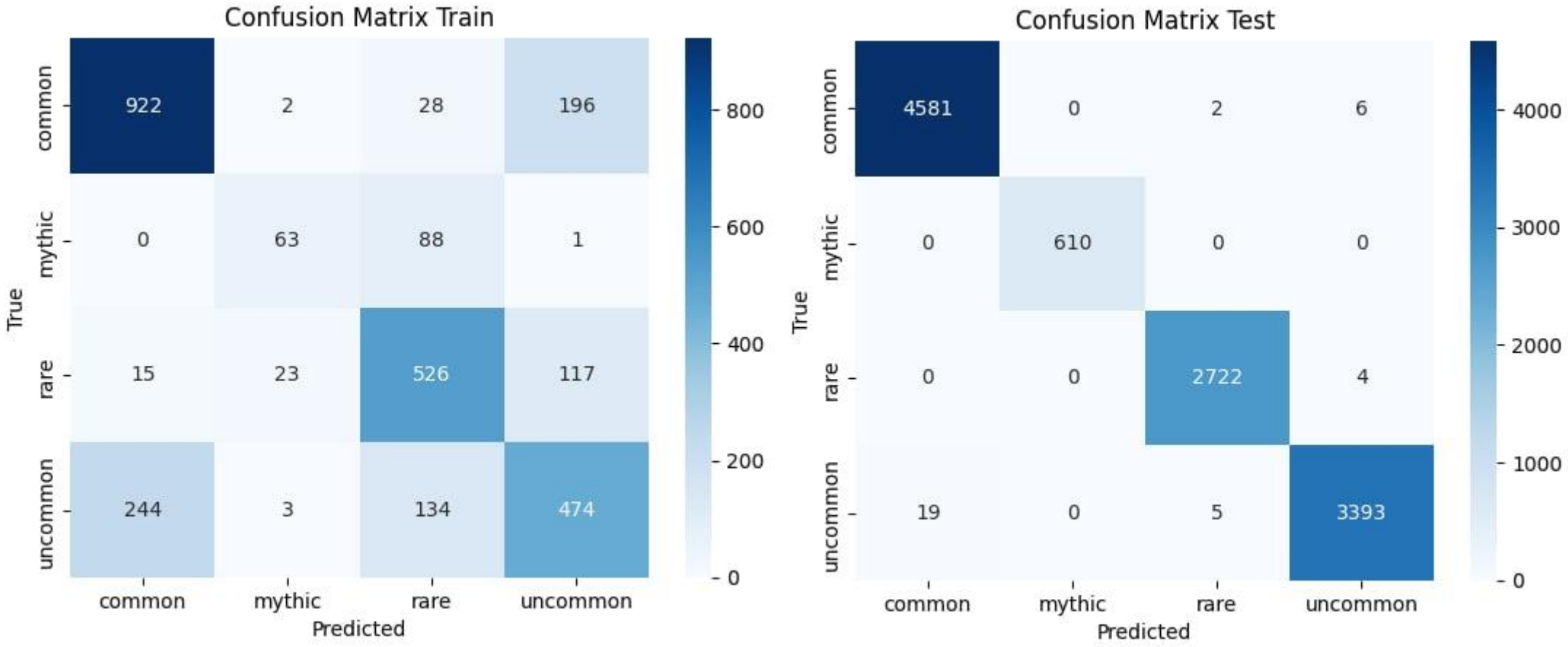
Loss Function: Multiclass LogLoss
Optimizer: ADAM
Accuracy:

- Test: 0.539
- Train: 0.979

Feature Importance – XGBoost Rarity



Confusion Matricies – XGBoost Rarity



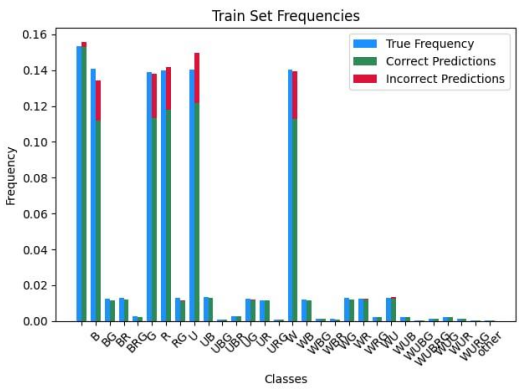
XGBoost – Color Classifcation

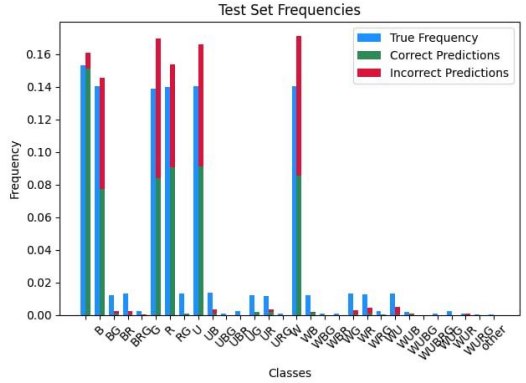
```
XGBoost_Parameters = {
        'n_estimators'     : 800,
        'max_depth'        : 3,
        'lr'               : 0.3,
        'subsample'        : 0.8,
        'colsample_bytree': 0.8,
        'reg_alpha'        : 0.5,
        'reg_lambda'       : 1.0}
```

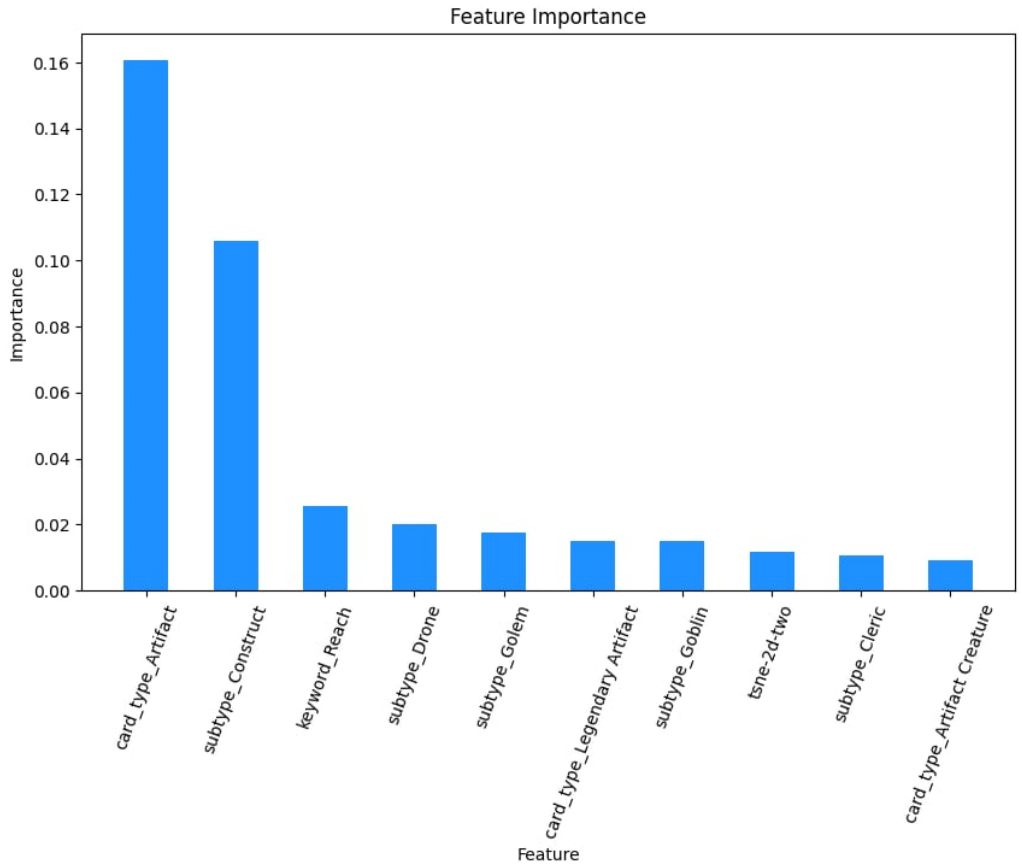
Loss function: Multiclass LogLoss

Model Evaluation

- Test accuracy: 59.12 %
- Train accuracy: 86.90 %







XGBoost - Converted Mana Cost

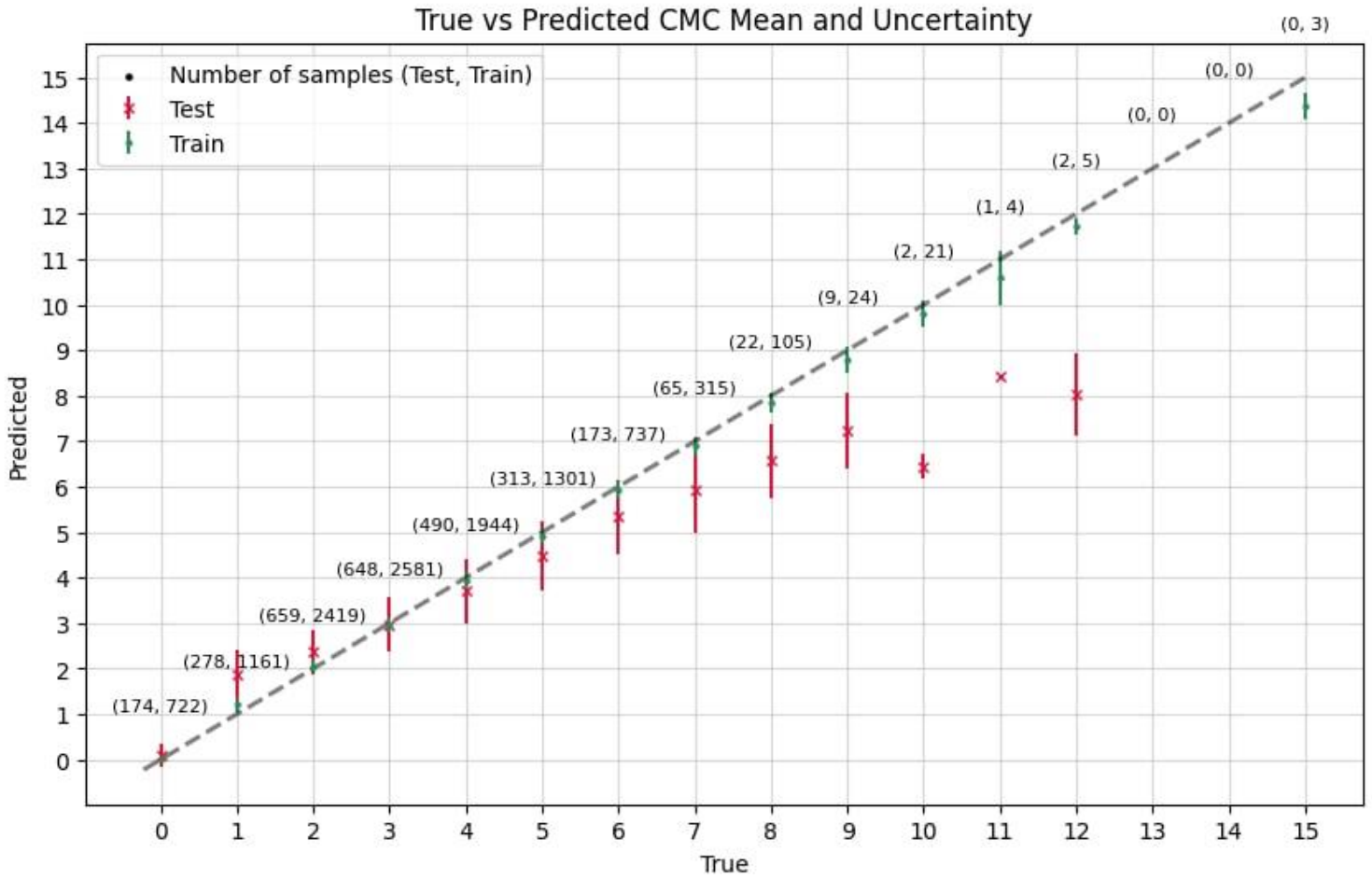
```
XGBoost_Parameters = {
        'n_estimators'     : 600,
        'max_depth'        : 10 ,
        'lr'               : 0.1,
        'subsample'        : 0.8,
        'colsample_bytree': 0.5,
        'gamma'            : 0.4,
        'reg_alpha'        : 0.4,
        'reg_lambda'       : 0.6}
```

Loss Function: MSE
Optimizer: ADAM
Accuracy:

- Test: 0.539
- Train: 0.979

R2-score:

- Test: 0.79
- Train: 0.99

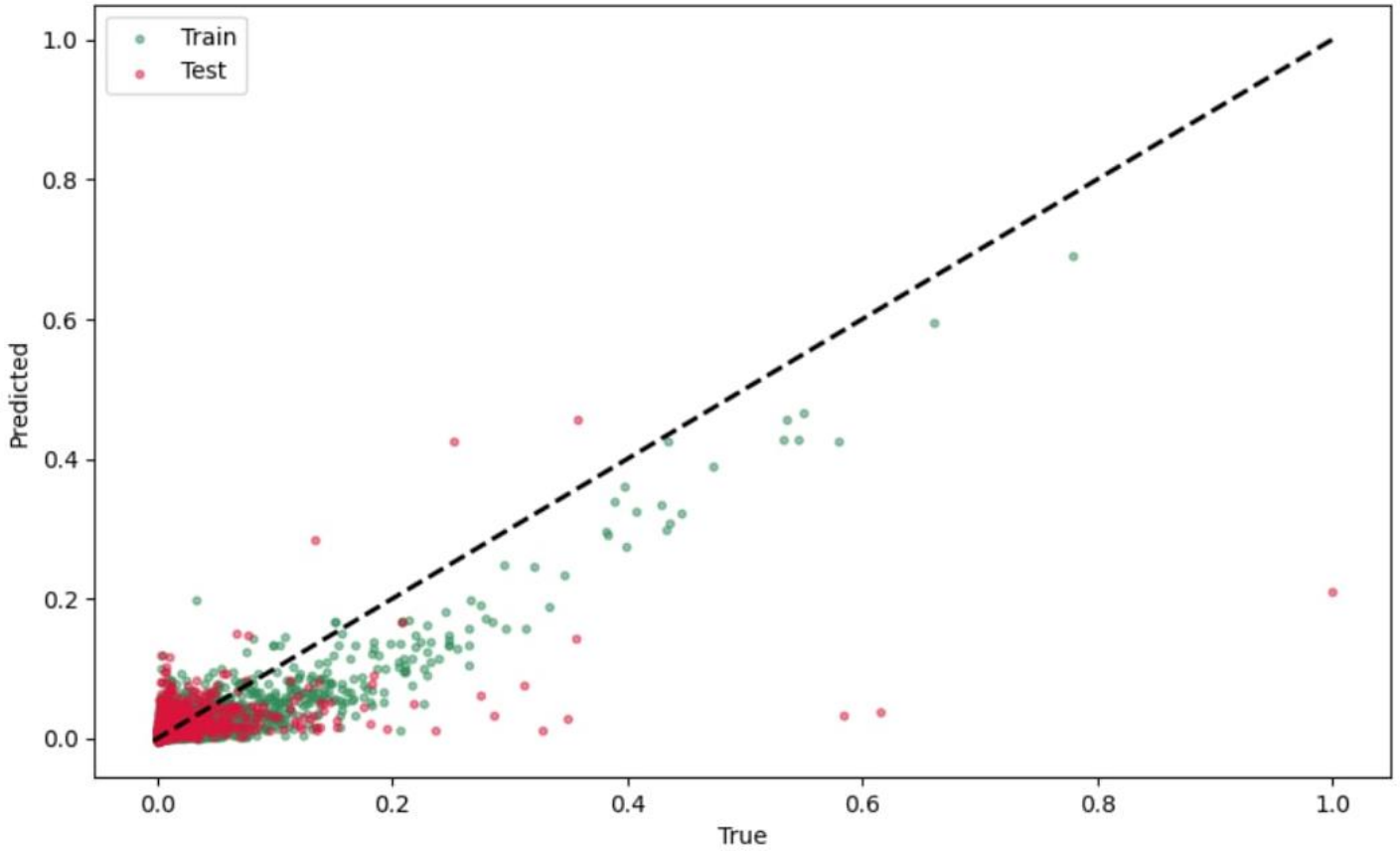


XGBoost – Card Prize

Bayesian Optimization:

```
XGBoost_bounds = {
        'n_estimators'     : (100,500),
        'max_depth'        : (3,10) ,
        'lr'               : (001,0.3),
        'subsample'        : (0.5,1.0),
        'colsample_bytree' : (0.3,0.9),
        'gamma'              : (0,5),
        'reg_alpha'        : (0.1,1),
        'reg_lambda'       : (0.1,1)}

XGBoost_Parameters = {
        'n_estimators'     : 433,
        'max_depth'        : 8 ,
        'lr'               : 0.13,
        'subsample'        : 0.8,
        'colsample_bytree' : 0.54,
        'gamma'              : 0.017,
        'reg_alpha'        : 0.2,
        'reg_lambda'       : 0.2}
```



Test R2 score 0.2
Removed Outliers R2 score: 0.3

,

Train R2 score: 0.82
Removed Outliers: 0.89

CNN – Multi Build

- Input: **Image (256,256,3) shape**
- Structure: (**CONV → CONV → POOL)x3 → DENSEx2 → SOFTMAX**
- Optimizer: **ADAM**
- Kernel Size: **(3, 3)**, Pool Size: **(2, 2)**
- Loss Rarity: **Sparse Categorical Cross Entropy**
- Loss Color: **Binary Cross Entropy** (One-Hot)
- Output: **Rarity and Color**
- Bayesian Optimization:

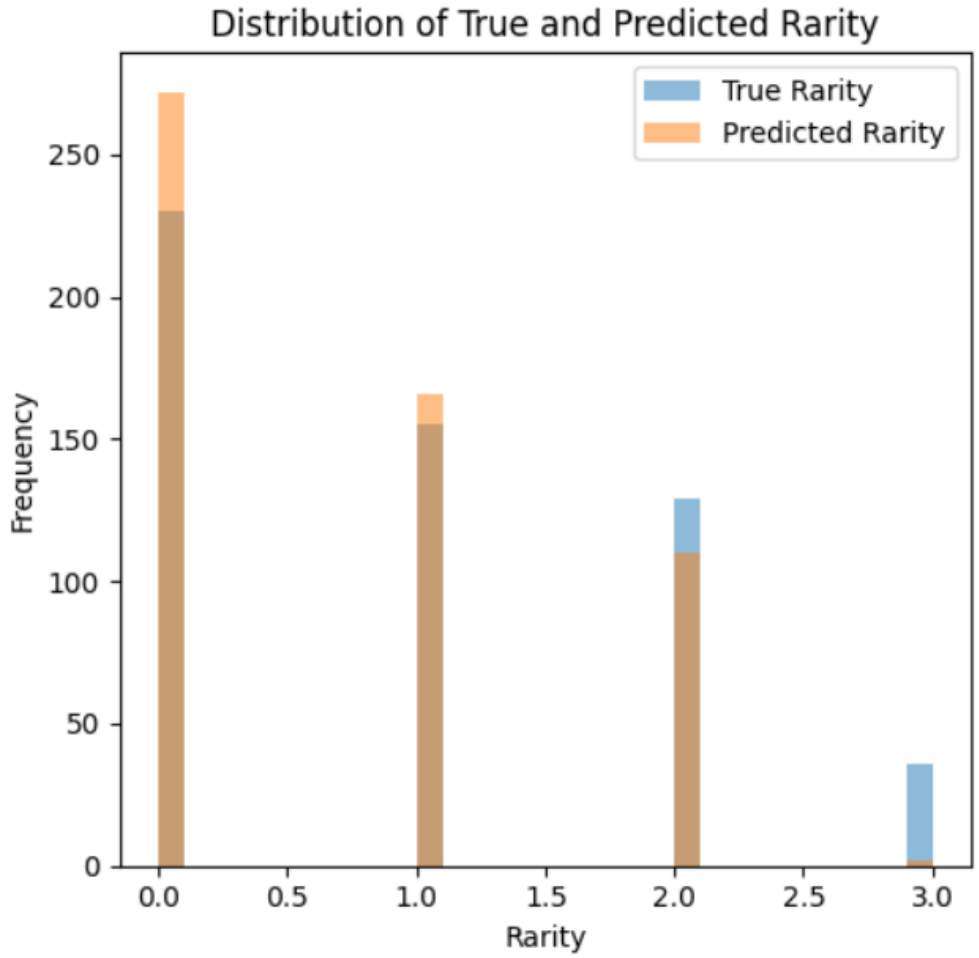
Color:
- Train Accuracy: 0.424
- Test Accuracy: 0.373

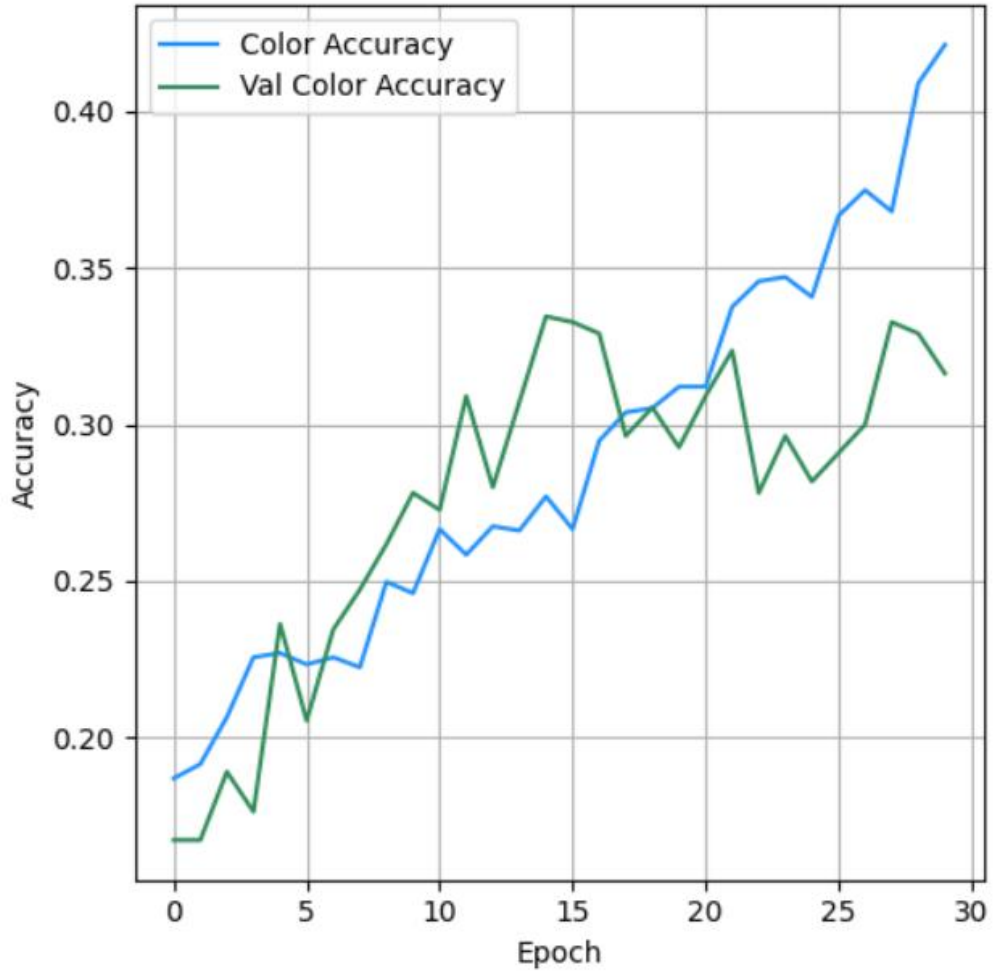
Rarity:
- Train Accuracy: 0.894
- Test Accuracy: 0.342

```
hyperparameter_space = {
    'conv_layer1' : (32, 128),
    'dense_size'  : (32, 256),
    'lr'          : (0.0001, 0.01)}
```

```
best_hyperparameter = {
    'conv_layer1' : 64,
    'dense_size'  : 196,
    'lr'          : 0.0001}
```

CNN – Multi Results





Combined image + Text (unoptimized)

Image Classifier

```python
def create_cnn_model():
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Text Classifier

```python
# Create and train the XGBoost model
model = xgb.XGBClassifier(
    verbosity=0,
    random_state=42,
    n_estimators=600,
    max_depth=5,
    learning_rate=0.1,
    reg_alpha=5.5,
    reg_lambda=5.5,
    subsample=0.8,
    colsample_bytree=0.8,
    scale_pos_weight=scale_pos_weight,
    early_stopping_rounds=10
)
model.fit(X_train_res, y_train_res, eval_set=[(X_test, y_test)], verbose=False)
```

Combined Classifier

```python
# Final model to combine XGBoost and CNN predictions
final_model = xgb.XGBClassifier(
    verbosity=0,
    random_state=42,
    n_estimators=800,
    max_depth=5,
    learning_rate=0.1,
    reg_alpha=10.5,
    reg_lambda=10.5,
    subsample=0.8,
    colsample_bytree=0.8,
)
```