In [160…
```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
import seaborn as sns
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split
```

EDA begins with importing the csv file as a pandas dataframe. This dataframe was then analyzed according to the provided data dictionary to determine the appropriate path to move forward. According to the data dictionary, it was given that all columns contained integer or continious numeric variables other than the LeagueIndex which was representative of the various ranks in Starcraft.

In [161…
```
playerdata = pd.read_csv("starcraft_player_data.csv")
```

In [162…
```
playerdata.set_index("GameID")
```

Out[162]:

GameID	LeagueIndex	Age	HoursPerWeek	TotalHours	APM	SelectByHotkeys	Assign
52	5	27	10	3000	143.7180	0.003515	
55	5	23	10	5000	129.2322	0.003304	
56	4	30	10	200	69.9612	0.001101	
57	3	19	20	400	107.6016	0.001034	
58	3	32	10	500	122.8908	0.001136	
...	...	...	...	...	...	...	
10089	8	?	?	?	259.6296	0.020425	
10090	8	?	?	?	314.6700	0.028043	
10092	8	?	?	?	299.4282	0.028341	
10094	8	?	?	?	375.8664	0.036436	
10095	8	?	?	?	348.3576	0.029855	

3395 rows × 19 columns

In [163…

```
data
```

Out[163]:

	GameID	LeagueIndex	Age	HoursPerWeek	TotalHours	APM	SelectByHotkeys
0	52	5	27	10	3000	143.7180	0.003515
1	55	5	23	10	5000	129.2322	0.003304
2	56	4	30	10	200	69.9612	0.001101
3	57	3	19	20	400	107.6016	0.001034
4	58	3	32	10	500	122.8908	0.001136
...	...	...	...	...	...	...	...
3335	9261	4	20	8	400	158.1390	0.013829
3336	9264	5	16	56	1500	186.1320	0.006951
3337	9265	4	21	8	100	121.6992	0.002956
3338	9270	3	20	28	400	134.2848	0.005424
3339	9271	4	22	6	400	88.8246	0.000844

3338 rows × 20 columns

ETL: With the EDA understanding, the next step was to locate any null values in the dataset. Although dropna did not result in any changes, there were '?' in the dataset that were functionally identical to a null value.

In [164…
```
data[data['Age']=='?']
```

Out[164]:

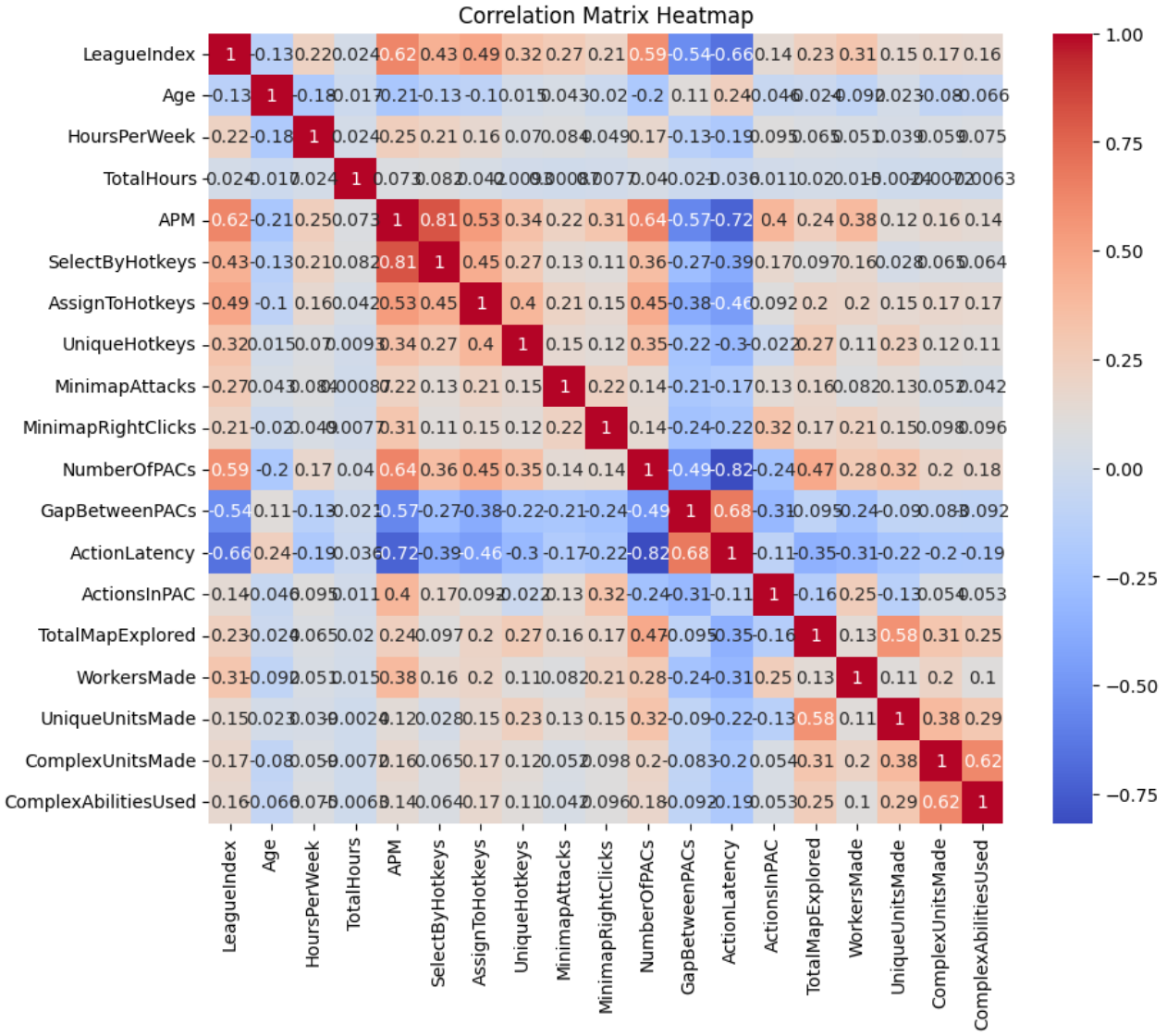
GameID	LeagueIndex	Age	HoursPerWeek	TotalHours	APM	SelectByHotkeys	AssignTo

In [165…
```
data.replace('?', np.nan, inplace=True)
data=data.dropna()
data['Age'] = pd.to_numeric(data['Age'], errors='coerce')
data['HoursPerWeek'] = pd.to_numeric(data['HoursPerWeek'], errors='coerce')
data['TotalHours'] = pd.to_numeric(data['TotalHours'], errors='coerce')
```

Further examination into the column found that the columns were not numeric and were then adjusted accordingly for the model.

In [166…
```
# Calculate correlation matrix
corr_matrix = data.drop(['GameID'], axis=1).corr()

# Plot correlation matrix heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
```



For the modeling, the first step was to realize that the goal or dependent variable being targeted was a categorical variable represented in integers. The predictor columns were placed in a correlation matrix to find and reduce collinearity. Although collinearity was found with the APM and Action Latency, both predictors were indicated to have significant correlation with the target variable. Significance of valeus were also tested and found that the columns of UniqueUnitsMade, ComplexUnitsMade, and COmplexAbilitiesUsed were not important to the model. However, with a DNN, the model will learn from the data and identify the less useful predictors. Therefore, it was decided that all of the predictors will be used for a deep neural network consisting of softmax activation function for classification.

In [172…

```
X = data.drop(['GameID','LeagueIndex'], axis=1)
y = data['LeagueIndex']

# Scale the features
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

#training sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0


# Define the model architecture
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(8, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metr

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=50, verbose=1)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')
```

```
Epoch 1/50
54/54 [==============================] - 1s 2ms/step - loss: 1.9013 - accura
cy: 0.2015
Epoch 2/50
54/54 [==============================] - 0s 3ms/step - loss: 1.6408 - accura
cy: 0.3176
Epoch 3/50
54/54 [==============================] - 0s 3ms/step - loss: 1.4648 - accura
cy: 0.3794
Epoch 4/50
54/54 [==============================] - 0s 2ms/step - loss: 1.3995 - accura
cy: 0.3993
Epoch 5/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3818 - accura
cy: 0.3831
Epoch 6/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3564 - accura
cy: 0.4086
Epoch 7/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3481 - accura
cy: 0.4082
Epoch 8/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3461 - accura
cy: 0.4086
```

```
Epoch 9/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3390 - accura
cy: 0.4097
Epoch 10/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3317 - accura
cy: 0.4236
Epoch 11/50
54/54 [==============================] - 0s 2ms/step - loss: 1.3392 - accura
cy: 0.3970
Epoch 12/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3253 - accura
cy: 0.4199
Epoch 13/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3207 - accura
cy: 0.4116
Epoch 14/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3160 - accura
cy: 0.4210
Epoch 15/50
54/54 [==============================] - 0s 2ms/step - loss: 1.3101 - accura
cy: 0.4221
Epoch 16/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3140 - accura
cy: 0.4199
Epoch 17/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3124 - accura
cy: 0.4236
Epoch 18/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3086 - accura
cy: 0.4300
Epoch 19/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3041 - accura
cy: 0.4210
Epoch 20/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3044 - accura
cy: 0.4146
Epoch 21/50
54/54 [==============================] - 0s 3ms/step - loss: 1.3011 - accura
cy: 0.4225
Epoch 22/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2969 - accura
cy: 0.4258
Epoch 23/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2961 - accura
cy: 0.4300
Epoch 24/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2939 - accura
cy: 0.4258
Epoch 25/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2935 - accura
cy: 0.4367
Epoch 26/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2949 - accura
```

```
cy: 0.4281
Epoch 27/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2886 - accura
cy: 0.4337
Epoch 28/50
54/54 [==============================] - 0s 2ms/step - loss: 1.2921 - accura
cy: 0.4288
Epoch 29/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2879 - accura
cy: 0.4322
Epoch 30/50
54/54 [==============================] - 0s 2ms/step - loss: 1.2895 - accura
cy: 0.4303
Epoch 31/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2837 - accura
cy: 0.4311
Epoch 32/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2802 - accura
cy: 0.4281
Epoch 33/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2829 - accura
cy: 0.4356
Epoch 34/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2794 - accura
cy: 0.4300
Epoch 35/50
54/54 [==============================] - 0s 2ms/step - loss: 1.2745 - accura
cy: 0.4404
Epoch 36/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2701 - accura
cy: 0.4427
Epoch 37/50
54/54 [==============================] - 0s 2ms/step - loss: 1.2755 - accura
cy: 0.4333
Epoch 38/50
54/54 [==============================] - 0s 2ms/step - loss: 1.2737 - accura
cy: 0.4416
Epoch 39/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2782 - accura
cy: 0.4363
Epoch 40/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2732 - accura
cy: 0.4303
Epoch 41/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2747 - accura
cy: 0.4348
Epoch 42/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2693 - accura
cy: 0.4423
Epoch 43/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2693 - accura
cy: 0.4449
Epoch 44/50
```

```
54/54 [==============================] - 0s 3ms/step - loss: 1.2717 - accura
cy: 0.4427
Epoch 45/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2618 - accura
cy: 0.4397
Epoch 46/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2627 - accura
cy: 0.4498
Epoch 47/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2598 - accura
cy: 0.4491
Epoch 48/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2614 - accura
cy: 0.4457
Epoch 49/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2599 - accura
cy: 0.4464
Epoch 50/50
54/54 [==============================] - 0s 3ms/step - loss: 1.2528 - accura
cy: 0.4498
Test Loss: 1.3547
Test Accuracy: 0.3982
```

Multiple models were attempted including KNN, random forest, simple, decision trees,and even linear regression to find the optimal classification model. Accuracy rates were found to be in the 0.30s for all models with DNN having around 0.4 for the test accuracy as the highest value. Therefore, deep neural network model was used and evaluated with sparse categorical crossentropy loss function to determine effectiveness in the test set.

INTERPRETATION

With the given metrics/predictors we can determine the rank of a starcraft player with an estimated 40% accuracy. Many of the predictors are representative of each other because values such as APM and unitsmade can be very closely related but still have a valuable impact on the rank of the player.

HYPOTHETICAL

Some steps to take to collect more informative data based on the EDA and model are to collect data across multiple seasons/years, gather a balanced representation of each rank to provide the learning model with better training sets, collect data on how each player in each game performs against the other player based on rank.