Data Preparation for EDA and Modeling

During the data preparation, I've made a few cleaning decisions:

1. In the player data, I want to capture data for popular pro players such as DRG and Vanya who have changed their names
2. I added the popular player Florencio to the players list and removed players who have names too similar to other words commonly used such as "has","zero", and "ready"
3. There are several numbers I want to keep which involves computer parts, years, matchups (1 vs 1, 2 vs 2) or numbers found in certain video games such as Cyberpunk 2077
4. I refined the stop words list to exclude words I believe would help grab a viewers attention, particularly words used when asking questions
5. I decided to keep words that are fully capitalized which I believe also help grab a viewers attention

In [1]:
```
# Imports
import pickle
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib.ticker as ticker
import matplotlib.image as mpimg
import seaborn as sns
import nltk
from nltk.corpus import stopwords
import string
import re
from nltk import word_tokenize, FreqDist
from sklearn.feature_extraction.text import CountVectorizer

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

import warnings
warnings.filterwarnings("ignore")

from Functions import *
```

Importing data pulled from APIs

In [8]:
```
all_stats = pickle.load(open(r"Data/all_stats.p","rb"))
players = pickle.load(open(r"Data/players_df.pickle","rb"))
```

In [9]:
```
# converting all_stats info a dataframe
main_df = pd.DataFrame.from_dict(all_stats).apply(lambda x: get_stats(x), axis=1).apply(pd.Series)
```

In [10]:
```
main_df['views'] = main_df['views'].astype(int)
main_df['likes'] = main_df['likes'].astype(int)
main_df['minutes'] = main_df['duration'].apply(lambda x: dur_to_min(x))
```

In [12]:
```
# Only keeping the last 3 years of YouTube video data
current_df = main_df.loc[main_df['date']>='2018']
```

Cleaning the Player Data

In [14]:
```
# Converting string data to lowercase
players['country'] = players['country'].apply(lambda x: str(x).lower())
players['race'] = players['race'].apply(lambda x: str(x).lower())
players['tag'] = players['tag'].apply(lambda x: str(x).lower())
```

In [15]:
```
# Removing player names too similar to other commonly used words in titles
remove = ['jim','punk','thor','hyperion','probe','control','alpha','fenix','golden','fear','flood','strange',
          'terran','zerg','scv','nexus','reaper','meat','blink','chance','mechanics','wave','next','nice','zero',
          'shadow','raise','job','doctor','time','has','phoenix','raise','sortof','dns','keen',
          'cham','prototype','academy','ranger','blacksmith','faith','eternity','chase','crimson',
          'albion','fate','tears','coffee','monster','ready','hunter','ling','turn','master','risky']
```

In [16]:
```
dropped_players = players.loc[~players['tag'].isin(remove)]
```

In [17]:
```
dropped_players.head()
```

Out[17]:

	country	id	race	resource_uri	tag
0	fi	485	z	/api/v1/player/485/	serral
1	kr	49	t	/api/v1/player/49/	maru
2	fr	5878	t	/api/v1/player/5878/	clem
3	it	5414	z	/api/v1/player/5414/	reynor
4	kr	76	z	/api/v1/player/76/	dark

In [18]:
```
player_cleaning_df = current_df.copy()
```

Adding labels for players in title

In [20]:
```
player_list = dropped_players.tag.values
```

In [21]:
```
# Creates a column with a list of players found in title
player_cleaning_df['players'] = player_cleaning_df['title'].apply(lambda x: get_from_list(x,player_list))
```

In [22]:
```
# Creates a column for each player and add 1 if player is in title
for i in range(len(player_list)):
    for player in player_cleaning_df.iloc[i].players:
        player_cleaning_df.at[i,player]=1
```

Finding average views and number of videos for each player

In [23]:
```
avg_views = []
count_views = []

# Find average views and number of videos, 0 if none
for player in player_list:
    try:
        avg_views.append(player_cleaning_df.loc[player_cleaning_df[player]==True].views.mean())
        count_views.append(player_cleaning_df.loc[player_cleaning_df[player]==True].views.count())
    except:
```

```
        avg_views.append(0)
        count_views.append(0)
```

In [24]:
```
# Add these values to the player info data frame
dropped_players['avg_views'] = avg_views
dropped_players['num_videos'] = count_views
```

In [25]:
```
dropped_players.head()
```

Out[25]:

	country	id	race	resource_uri	tag	avg_views	num_videos
0	fi	485	z	/api/v1/player/485/	serral	160802.518519	54
1	kr	49	t	/api/v1/player/49/	maru	145652.972973	37
2	fr	5878	t	/api/v1/player/5878/	clem	146513.170213	47
3	it	5414	z	/api/v1/player/5414/	reynor	144567.088889	45
4	kr	76	z	/api/v1/player/76/	dark	129995.939394	33

Getting Country Data

In [26]:
```
countries_list = np.array(list(set(dropped_players.country)))
```

In [28]:
```
countries_list.astype(str)
```

Out[28]:
```
array(['kr', 'vn', 'si', 'id', 'lv', 'tr', 'au', 'cu', 'jp', 'lu', 'ua',
       'nl', 'dz', 'gt', 'us', 'be', 'ie', 'ee', 'pe', 'bo', 've', 'co',
       'cr', 'uz', 'ba', 'ca', 'cz', 'cn', 'at', 'tw', 'ph', 'nz', 'ch',
       'il', 'br', 'ro', 'es', 'hr', 'no', 'bg', 'se', 'mx', 'hk', 'uk',
       'hu', 'sg', 'dk', 'it', 'none', 'fi', 'sk', 'za', 'my', 'eg', 'ar',
       'mm', 'bd', 'de', 'lt', 'ru', 'pl', 'fr', 'in', 'is', 'cl', 'by'],
      dtype='<U4')
```

In [29]:
```
# Adds list of countries associated with the player names found in titles
player_cleaning_df['country'] = player_cleaning_df['title'].apply(
                                lambda x: get_player_info(x,player_list,dropped_players,info='country'))
```

In [30]:
```
for i in range(len(player_cleaning_df)):
    for country in player_cleaning_df.iloc[i].country:
        player_cleaning_df.at[i,country]=1
player_cleaning_df.fillna(0,inplace=True)
```

In [31]:
```
avg_views_country = []
count_views_country = []

# Find average views and number of videos for countries
for country in countries_list:
    try:
        avg_views_country.append(player_cleaning_df.loc[player_cleaning_df[country]==1].views.mean())
        count_views_country.append(player_cleaning_df.loc[player_cleaning_df[country]==1].views.count())
    except:
        avg_views_country.append(0)
        count_views_country.append(0)
```

In [32]:
```
countries_df = pd.DataFrame(data=[countries_list])
```

In [33]:
```
countries_df = countries_df.T
```

In [34]:
```
countries_df['avg_views'] = avg_views_country
countries_df['num_videos'] = count_views_country
```

In [35]:
```
countries_df.columns = ['country_code','avg_views','num_videos']
```

In [37]:
```
# imports csv with country codes and names
ccodes = pd.read_csv(r'Data\country_codes.csv',encoding='latin-1')
ccodes['Code'] = ccodes['Code'].apply(lambda x: str(x).lower().strip())
```

In [38]:
```
country_zip = countries_df['country_code']
```

In [39]:
```
country_container = []
for country in country_zip:
    try:
        country_container.append(ccodes.loc[ccodes['Code']==country,'Country'].values[0])
    except:
        country_container.append("None")
```

In [40]:
```
countries_df['country'] = country_container
```

In [41]:
```
countries_df.head()
```

Out[41]:

	country_code	avg_views	num_videos	country
0	kr	118722.110619	226	Korea, Republic of
1	vn	0.000000	0	Viet Nam
2	si	0.000000	0	Slovenia
3	id	0.000000	0	Indonesia
4	lv	0.000000	0	Latvia

Getting SC2 Race Data

In [42]:
```
# Following similar logic to the countries, except the totals will have number of races instead of just a label
```

In [43]:
```
# Starcraft 2 races = Zerg, Protoss, Terran
sc2races = ['z','p','t']
```

In [44]:
```
# Adds list of countries associated with the player names found in titles
player_cleaning_df['sc2race'] = player_cleaning_df['title'].apply(
                                lambda x: get_player_info(x,player_list,dropped_players,info='race'))
player_cleaning_df['z'] = 0
player_cleaning_df['t'] = 0
player_cleaning_df['p'] = 0
```

In [46]:
```
for i in range(len(player_cleaning_df)):
    for race in player_cleaning_df.iloc[i].sc2race:
        try:
            player_cleaning_df.at[i,race]+=1
```

```
        except:
            player_cleaning_df.at[i,race]=1
```

In [47]:
```
avg_views_race = []
count_views_race = []

# Find average views and number of videos for sc2 races
for race in sc2races:
    try:
        avg_views_race.append(player_cleaning_df.loc[player_cleaning_df[race]>0].views.mean())
        count_views_race.append(player_cleaning_df.loc[player_cleaning_df[race]>0].views.count())
    except:
        avg_views_race.append(0)
        count_views_race.append(0)
```

In [48]:
```
sc2races_df = pd.DataFrame(data=[sc2races])
sc2races_df = sc2races_df.T
```

In [49]:
```
sc2races_df['avg_views'] = avg_views_race
sc2races_df['num_videos'] = count_views_race
sc2races_df.columns = ['race','avg_views','num_videos']
```

In [50]:
```
sc2races_df
```

Out[50]:

	race	avg_views	num_videos
0	z	133687.225941	239
1	p	119526.383333	180
2	t	129004.323077	195

Getting Matchup Data

- Focusing on just the 1 vs 1 matchups for this EDA, which is the most popular in this channel

In [51]:
```
# 6 combinations of matchups for 3 races
sc2matchups = ['ZvZ','ZvP','ZvT','TvT','TvP','PvP']
```

In [52]:
```
for matchup in sc2matchups:
    player_cleaning_df[matchup] = 0
```

In [53]:
```
# Assigns a label for each combination of matchup
for i in range(len(player_cleaning_df)):
    race = player_cleaning_df.iloc[i].sc2race
    if len(race) == 2:
        if ("z" in race) & ("t" in race):
            player_cleaning_df.at[i,'ZvT']=1
        elif ("z" in race) & ("p" in race):
            player_cleaning_df.at[i,'ZvP']=1
        elif ("t" in race) & ("p" in race):
            player_cleaning_df.at[i,'TvP']=1
        elif ("z" in race):
            player_cleaning_df.at[i,'ZvZ']=1
        elif ("t" in race):
```

```
            player_cleaning_df.at[i,'TvT']=1
        elif ("p" in race):
            player_cleaning_df.at[i,'PvP']=1
```

In [54]:
```
matchup_df = pd.DataFrame(data=[sc2matchups])
matchup_df = matchup_df.T
```

In [55]:
```
avg_views_matchup = []
count_views_matchup = []

# Find average views and number of videos for sc2 races
for matchup in sc2matchups:
    try:
        avg_views_matchup.append(player_cleaning_df.loc[player_cleaning_df[matchup]>0].views.mean())
        count_views_matchup.append(player_cleaning_df.loc[player_cleaning_df[matchup]>0].views.count())
    except:
        avg_views_matchup.append(0)
        count_views_matchup.append(0)
```

In [56]:
```
matchup_df['avg_views'] = avg_views_matchup
matchup_df['num_videos'] = count_views_matchup
matchup_df.columns = ['matchup','avg_views','num_videos']
```

In [57]:
```
matchup_df
```

Out[57]:

	matchup	avg_views	num_videos
0	ZvZ	119969.833333	24
1	ZvP	138716.631579	57
2	ZvT	148570.689655	87
3	TvT	93882.444444	18
4	TvP	115359.255319	47
5	PvP	71545.333333	18

Getting Game Data

- Game titles are usually before the ":" in the titles

In [58]:
```
def game_split(message):
    if ":" in message:
        return (game_cleaner("".join((message.lower().split(":")[0].split()))))
```

In [59]:
```
player_cleaning_df['game'] = player_cleaning_df['title'].apply(lambda x:game_split(x))
player_cleaning_df['game'].fillna("none",inplace=True)
```

In [61]:
```
player_cleaning_df['game'].value_counts()
```

Out[61]:
```
starcraft2    978
none          119
warcraft3      49
frostpunk      40
zelda          11
```

```
ageofempires4              11
starcraft                  11
sekiro                      8
underlords                  4
theyarebillions             4
worldofwarcraft             3
hearthstone                 2
tropico6                    1
sabzu                       1
arise                       1
aplaguetale                 1
fortnite                    1
thestanleyparable           1
farcry5                     1
darksouls3                  1
ironharvest                 1
diablo4announcement         1
mutantyearzero              1
worldofwarcraftclassic      1
Name: game, dtype: int64
```

In [62]:
```
player_cleaning_df['game'].value_counts()
```

Out[62]:
```
starcraft2                978
none                      119
warcraft3                  49
frostpunk                  40
zelda                      11
ageofempires4              11
starcraft                  11
sekiro                      8
underlords                  4
theyarebillions             4
worldofwarcraft             3
hearthstone                 2
tropico6                    1
sabzu                       1
arise                       1
aplaguetale                 1
fortnite                    1
thestanleyparable           1
farcry5                     1
darksouls3                  1
ironharvest                 1
diablo4announcement         1
mutantyearzero              1
worldofwarcraftclassic      1
Name: game, dtype: int64
```

In [63]:
```
game_df = player_cleaning_df[['game','views']].groupby('game').agg(['mean','count']).reset_index()
game_df.columns = ['game_title','avg_views','num_videos']
```

Word Count Tokenizer

In [65]:
```
# These stop words don't add much to the analysis.  Kept other common stop words
special_stops = ['of','OF','is','the','THE','by','BY','it','in','on','and','but','being','an','for','to','they','any','from',
                 'then','some','you','your','their','as','about','out','with','his','hers','he','she','at','go']
```

One Word Tokenizer

In [66]:
```
one_words = CountVectorizer(preprocessor=splitter,stop_words=special_stops)
one_words_counter = one_words.fit_transform(player_cleaning_df['title'])
```

Bi/Tri gram Tokenizer

In [69]:
```
# Using a CountVectorizer with the predefined splitter function and special stop words for processing
# Also tokenizing for tri-grams to catch matchups X vs Y
grammed = CountVectorizer(preprocessor=splitter,stop_words=special_stops,ngram_range=(1,3))
grammedcounter = grammed.fit_transform(player_cleaning_df['title'])
```

In [71]:
```
len(grammed.vocabulary_)
```

Out[71]:
```
9805
```

In [72]:
```
grammed_df = pd.DataFrame(data=grammedcounter.toarray(),columns=grammed.get_feature_names())
grammed_df['views'] = player_cleaning_df['views']
```

In [76]:
```
highest_mean = []
for col in grammed_df.columns:
    highest_mean.append([col,grammed_df['views'].loc[grammed_df[col]==1].mean()])
```

Data Exports

In [78]:
```
# # Players_cleaning_df for use with visualizations
# with open(r'Data\players_cleaned_df.pickle', 'wb') as f:
#     pickle.dump(player_cleaning_df, f)

# # 1gram - Trigram Tokenizers  grammed, one_words
# with open(r'Data\one_words.pickle','wb') as f2:
#     pickle.dump(one_words,f2)
# with open(r'Data\grammed.pickle','wb') as f3:
#     pickle.dump(grammed,f3)
# with open(r'Data\grammed_df.pickle','wb') as f3_2:
#     pickle.dump(grammed_df,f3_2)

# # Matchups
# with open(r'Data\matchup_df.pickle','wb') as f4:
#     pickle.dump(matchup_df,f4)

# # Games
# with open(r'Data\game_df.pickle','wb') as f5:
#     pickle.dump(game_df,f5)

# # StarCraft 2 races
# with open(r'Data\sc2races_df.pickle','wb') as f6:
#     pickle.dump(sc2races_df,f6)

# # Countries
# with open(r'Data\countries_df.pickle','wb') as f7:
#     pickle.dump(countries_df,f7)

# # Players
# with open(r'Data\player_df.pickle','wb') as f8:
#     pickle.dump(dropped_players,f8)
```