Imports

```
from collections import Counter
from google.colab import drive, files
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import pprint
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
from typing import List
import locale
locale.setlocale( locale.LC_ALL, 'en_US.UTF-8' );
```

Loading the Data

The dataset is located in the shared Google Drive folder and is named `database.jsonl.gz`. In order for Google Cola you must make sure the shared folder is a part of your personal Drive. To do this, navigate to the shared folder unde right click the folder, and select "Add to My Drive."

Then, run the cells below to mount your drive as a filesystem, and to load the dataset.

```
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/d
```

```
os.chdir("/content/drive/My Drive/Data Mining Shared Folder")
!ls -lah
```

```
total 13G
-rw------- 1 root root  151 Oct  3 01:21 'Course Project Moodle.gdoc'
-rw------- 1 root root 791K Dec 13 05:25 'Data Analysis.ipynb'
-rw------- 1 root root 113K Dec 13 02:39 'Data Analysis Local Connor.ipynb'
-rw------- 1 root root 705K Nov 30 20:56 'Data Analysis Local.ipynb'
-rw------- 1 root root  12G Oct 25 00:29  database.json
-rw------- 1 root root 1.2G Nov 16 19:43  database.jsonl.gz
-rw------- 1 root root 170M Nov 30 19:16  database_new.jsonl.gz
-rw------- 1 root root  12M Oct 25 18:01  database_sample.json
-rw------- 1 root root  151 Dec 13 05:01 'Data Mining Slides.gslides'
-rw------- 1 root root 668K Dec 13 05:24  df_labels.csv
-rw------- 1 root root  62M Dec 13 05:24  df_train.csv
-rw------- 1 root root  151 Nov  6 02:14  exampleForEachField.gdoc
-rw------- 1 root root 622K Nov  3 23:57  exampleForEachField.txt
-rw------- 1 root root  151 Dec 13 05:13 'Key Results.gdoc'
-rw------- 1 root root 3.4M Oct 25 00:16  screenlog.0
```

The code below acts as a sanity check, reading in the database and printing out each game and its associated app I

```
# Note, each time this object is consumed (iterated over) it needs to be re-defined if you want to read fr
```

```
# Note, each time this object is consumed (iterated over) it needs to be re-defined if you want to read it
dataset = pd.read_json("/content/drive/My Drive/Data Mining Shared Folder/database_new.jsonl.gz", orient="
```

```
# Sanity check - print out the app ID ad name of all the games in the database
chunk: pd.DataFrame
for chunk in dataset:
  print(chunk[["appid", "name"]])
```

```
       appid                       name
0       10.0             Counter-Strike
1       20.0      Team Fortress Classic
2       30.0               Day of Defeat
3       40.0         Deathmatch Classic
4       50.0  Half-Life: Opposing Force
..       ...                        ...
995  46490.0               Still Life 2
996  46500.0                    Syberia
997  46510.0                  Syberia 2
998  46520.0            Wasteland Angel
999  46540.0               Trapped Dead

[1000 rows x 2 columns]
         appid                                     name
1000   46550.0                              Post Mortem
1001   46560.0                               Robin Hood
1002   46570.0  Grotesque Tactics 2 - Dungeons and Donuts
1003   46600.0                              Swarm Arena
1004   46700.0              Ironclads: American Civil War
...        ...                                      ...
1995  236930.0                         Blackwell Epiphany
1996  236970.0             Jack Keane 2 - The Fire Within
1997  237110.0             Mortal Kombat Komplete Edition
1998  237310.0                                   Elsword
1999  237350.0                            Frozen Cortex

[1000 rows x 2 columns]
       appid                                            name
2000  237430                         Expeditions: Conquistador
2001  237470                             Battle Worlds: Kronos
2002  237550               Realms of Arkania: Blade of Destiny
2003  237570  Penny Arcade's On the Rain-Slick Precipice of ...
2004  237740                   Angry Video Game Nerd Adventures
...      ...                                              ...
2995  281390                                           O.R.B.
2996  281410                                  Ubersoldier II
2997  281430                                           Clans
2998  281450                     Disciples Sacred Lands Gold
2999  281560                                   D.W.A.R.F.S.

[1000 rows x 2 columns]
       appid                          name
3000  281580             Wings Over Europe
3001  281610  Homeworld: Deserts of Kharak
3002  281640             The Banner Saga 2
3003  281750                         Munin
3004  281820                   Explodemon
...      ...                           ...
3995  314610         Vincere Totus Astrum
3996  314630             The Thin Silence
3997  314650                  SpaceEngine
3998  314660      Oddworld: New 'n' Tasty
3999  314710                 Mighty No. 9

[1000 rows x 2 columns]
       appid                          name
4000  314760      Direct Hit: Missile War
4001  314770            Pe-2: Dive Bomber
4002  314790                      Silence
4003  314810              Randal's Monday
```

```
4004  314830               Blackguards 2
...      ...                        ...
4995  343270    Disillusions Manga Horror
4996  343280                      Zotrix
4997  343320  StaudSoft's Synthetic World
4998  343340                    Tiamat X
4999  343360                   Particula

[1000 rows x 2 columns]
       appid                        name
5000  343370                  SLAMMED!
5001  343390  Elementary My Dear Majesty!
5002  343410                    inSynch
5003  343430                   Hypnosis
5004  343440               Crash Drive 2
...      ...                        ...
5995  367700          SHOFER Race Driver
5996  367710            Afterlife Empire
5997  367780                Aero's Quest
5998  367800               Diamond Deeps
5999  367820                 Decromancer

[1000 rows x 2 columns]
       appid                         name
6000  367990                3D Paraglider
6001  368000             100ft Robot Golf
6002  368040                     ExoCorps
6003  368050  Almightree: The Last Dreamer
6004  368070       Sniper Ghost Warrior 3
...      ...                          ...
6995  394140                  Sound Shift
6996  394160   ARCADE GAME SERIES: PAC-MAN
6997  394220                Last Horizon
6998  394230                  Battleborn
6999  394260                 Dance Magic

[1000 rows x 2 columns]
       appid                             name
7000  394270                    Country Tales
7001  394280  Dark Heritage: Guardians of Hope
7002  394290               Tennis in the Face
7003  394310                      Punch Club
7004  394360               Hearts of Iron IV
...      ...                              ...
7995  423720                     Color Chaos
7996  423730                      Hyper Gods
7997  423740                Save Your Mother
7998  423750                     Gardenarium
7999  423760                    Hit Tank PRO

[1000 rows x 2 columns]
       appid                                name
8000  423770                 Our Love Will Grow
8001  423780                       Zero Gravity
8002  423800                  Automata Empire
8003  423810                        Marooners
8004  423870                   Astervoid 2000
...      ...                                ...
8995  451760                 Highway Blossoms
8996  451780                         Trillion
8997  451800                  End Of The Mine
8998  451840                      Out of Ammo
```

```
8999  451870  Nighttime Terror: Dessert Defender

[1000 rows x 2 columns]
       appid                                    name
9000  451880                   Catch a Falling Star
9001  451900                             WizardCraft
9002  451920               Thorne - Death Merchants
9003  451930                                 qrth-phyl
9004  451940                             Return Home
...      ...                                     ...
9995  486550                          The Caretaker
9996  486630                           Toxic Terror
9997  486640                  Cheaters Blackjack 21
9998  486650  Potato Thriller Steamed Potato Edition
9999  486660                          Super Mixtape

[1000 rows x 2 columns]
        appid                            name
10000  486690                Pastelia Stories
10001  486720                   Bastard Bonds
10002  486760               Master Of Marbles
10003  486780                  Fruit Ninja VR
10004  486810    House of Snark 6-in-1 Bundle
...       ...                             ...
10995  512230                     Sally's Law
10996  512240                    Duckpocalypse
10997  512250  Oh...Sir! The Insult Simulator
10998  512260      Avalon Legends Solitaire 2
10999  512270           Home - A VR Spacewalk

[1000 rows x 2 columns]
        appid                        name
11000  512300                      Unbox
11001  512370                   Hero Boy
11002  512410   69 Ways to Kill a Zombie
11003  512420              Wave Magic VR
11004  512430   Make America Great Again
...       ...                         ...
11995  539450                   Crab Dub
11996  539460                Puzzle Cube
11997  539470             Police Stories
11998  539560  Vienna Automobile Society
11999  539640                 Racecar.io

[1000 rows x 2 columns]
        appid                           name
12000  539650         Twelve Sky 2 Classic
12001  539660                   ChronoClock
12002  539670                   Sakura Nova
12003  539690                Nanomedix Inc
12004  539720               Razortron 2000
...       ...                            ...
12995  565200                   Gem Monster
12996  565330                       Twisted
12997  565380  Cargo Cult: Shoot'n'Loot VR
12998  565390                      Hyperun
12999  565490                       Tier 1

[1000 rows x 2 columns]
        appid                                                name
13000  565540                       The Turkey of Christmas Past
13001  565550                                    Redneck Rampage
```

```
13002  565600  Christmas Stories: Nutcracker Collector's Edition
13003  565640                                       Colorful Life
13004  565650                                               SWAM
...       ...                                                ...
13995  592620                                         Trajectory
13996  592640                                      MGSLeisure1000
13997  592660                                          Zen Garden
13998  592720                                        Paulo's Wing
13999  592730                                   Aerial Destruction

[1000 rows x 2 columns]
        appid                                               name
14000  592750                                        SPACE-FRIGHT
14001  592780                                        Bang Bang Car
14002  592890                                             Insidia
14003  592990             ChemCaper: Act I - Petticles in Peril
14004  593030             Strategic Command WWII: War in Europe
...       ...                                                ...
14995  621070                                 Legends of Ellaria
14996  621080                    Persian Nights: Sands of Wonders
14997  621090  Elixir of Immortality II: The League of Immort...
14998  621140                                       EGG HUNT VR
14999  621150                                     False Shelter

[1000 rows x 2 columns]
        appid                                          name
15000  621170          Winter's Empty Mask - Visual novel
15001  621220                                   Nantucket
15002  621290                                  The Hunted
15003  621780                       Virtually Impossible
15004  621810                             Strain Tactics
...       ...                                         ...
15995  654850               Masked Forces: Zombie Survival
15996  654880         Dream Daddy: A Dad Dating Simulator
15997  654890  Grand Tactician: The Civil War (1861-1865)
15998  654900                                         qop
15999  654910                             YANKAI'S PEAK.

[1000 rows x 2 columns]
        appid                    name
16000  654940                     RXE
16001  654950        Labyrinth Escape
16002  654960  The Eldritch Zookeeper
16003  654970            Numberline 2
16004  654980          Inferno Puzzle
...       ...                     ...
16995  676840  Contagion VR: Outbreak
16996  676850                Guardian
16997  676880   Tales of Glacier (VR)
16998  676910  Magic Potion Destroyer
16999  676930                 TwinCop

[1000 rows x 2 columns]
        appid                          name
17000  676960                 Hotlap Heroes
17001  676990                     The Stone
17002  677010                      Darts VR
17003  677020  Eventide 3: Legacy of Legends
17004  677030                    Draw Souls
...       ...                           ...
17995  701610         Dungeons of the Fallen
17996  701620                Monster partner
```

```
17997  701630                   Saint George
17998  701670                     krotruvink
17999  701680             The Legend of Slime

[1000 rows x 2 columns]
        appid                                          name
18000  701720                                 2D Neon Cube
18001  701730                     Yi and the Thousand Moons
18002  701740                                 Modest Kind
18003  701760                                       L.S.S
18004  701800                                  TRIPLICATA
...       ...                                          ...
18995  726460                         Choice of Broadsides
18996  726490                       Projection: First Light
18997  726500  The Adventures of Sam Carlisle: The Hunt for t...
18998  726510                                   Heads Run
18999  726520                         The Merchant Memoirs

[1000 rows x 2 columns]
        appid                        name
19000  726570               Heavy Destinies
19001  726580  Game Machines: Arcade Casino
19002  726590          NBA 2K Playgrounds 2
19003  726600                    Drift Zone
19004  726610                   Podium Bash
...       ...                          ...
19995  752600                    Dual Snake
19996  752640        National Rugby Manager
19997  752690                The Resistance
19998  752700                       Abscond
19999  752710                 Infinity Trip

[1000 rows x 2 columns]
        appid                                  name
20000  752720                         The Culling 2
20001  752730  Fantastic Beasts and Where to Find Them
20002  752750                    Justice League VR
20003  752760                   Isle Tower Defense
20004  752770                          GyroShooter
...       ...                                  ...
20995  777380                  Fleets of Ascendancy
20996  777400                  Ultimate Spider Hero
20997  777410                            EarthNight
20998  777420                    Quests Unlimited
20999  777430                 Virtual Boxing League

[1000 rows x 2 columns]
        appid                               name
21000  777530                     Salsa Virtual
21001  777540                Entertainment Hero
21002  777550                       King Battle
21003  777560                         INVISIBLE
21004  777580  Champ Against Chumps Upgrade Edition
...       ...                                ...
21995  801330                       Monkey Rush
21996  801340                    Healer Simulator
21997  801370                           Isotower
21998  801380                              MOVIT
21999  801390                     Onslaught VR

[1000 rows x 2 columns]
```

```
        appid                                               name
22000  801420                                        Kollidoskop!
22001  801450                                Legend Store Simulator
22002  801480                        Agent A: A puzzle in disguise
22003  801490                                          HellCrunch
22004  801500                                           WannaMine
...       ...                                                 ...
22995  826380  Dark Canvas: A Murder Exposed Collector's Edition
22996  826420  Dark Romance: Heart of the Beast Collector's E...
22997  826430  Twilight Phenomena: The Incredible Show Collec...
22998  826450                                    Paws 'n Claws VR
22999  826460                                   The Road to Hades

[1000 rows x 2 columns]
        appid                            name
23000  826480            VR Benchmark Kanojo
23001  826540                     Audio Trip
23002  826550                  Golf Cart Drive
23003  826580  Mervin and the Wicked Station
23004  826590                         Kmenta
...       ...                             ...
23995  852920                   Bad Bots Rise
23996  853000                  Generic Jumper
23997  853020        Venal Soul (Chapter One)
23998  853040  DEEP SPACE | Space-Platformer
23999  853050                        El Hijo

[1000 rows x 2 columns]
        appid                                      name
24000  853110                 Viki Spotter: Sports
24001  853120                          Iron Ground
24002  853140           Adeptus Titanicus: Dominus
24003  853150  Wars Across The World: Russian Battles
24004  853200      Shred! 2 - Freeride Mountainbiking
...       ...                                    ...
24995  877530                          Sneaky Funk
24996  877550                Digital Jigsaw Puzzle
24997  877590                         幻想郷ローリングフォース
24998  877680  Hardcore Weapon Challenge - FPS Action
24999  877730                       Deadly Silence

[1000 rows x 2 columns]
        appid                          name
25000  877750                        Coinon
25001  877780        Puzzles for smart: Cats
25002  877800                    Swimsanity!
25003  877810      Anodyne 2: Return to Dust
25004  877820  LOOT BOX ACHIEVEMENT SIMULATOR
...       ...                           ...
25995  905180                Inevitable VR
25996  905220                   Grim Earth
25997  905230                      IQ Test
25998  905240        Street Tuning Evolution
25999  905260                    Hags Castle

[1000 rows x 2 columns]
        appid                   name
26000  905270          Merlin Soccer
26001  905280                 QB Sim
26002  905300        Daytona Racing
26003  905330        oOo: Ascension
26004  905340              Heave Ho
```

```
...       ...                    ...
26995  931880              Tekling 2
26996  931930  The Haunted Graveyard
26997  931940        Junkyard Wizard
26998  932050                希望之星
26999  932070          七夜怪谈 都市校园禁锢传说

[1000 rows x 2 columns]
        appid                                              name
27000  932090                                           Triwave
27001  932120                                    TAL: Arctic 3
27002  932150                                             WWTF
27003  932160                                    Late at night
27004  932170                    Who Wants To Destroy An Alien
...       ...                                               ...
27995  958800                                           Lapse
27996  958880  Nick Bounty: The Dame with the Blue Chewed Shoe.
27997  958900                                 BATTLE X Arcade
27998  958910                                  AR-K: END GAME
27999  958930                                  Ricky Recharge

[1000 rows x 2 columns]
        appid                                              name
28000  958960                                     Darwin's Test
28001  958980                                       Pang & Bang
28002  958990                                            Blokin
28003  959000  Edna & Harvey: The Breakout - Anniversary Edition
28004  959020                             You Shall Not Break!
...       ...                                               ...
28995  986040                                   The Unliving
28996  986070                                    Spirit Oath
28997  986080                                Zombie Apocalypse
28998  986130                                 Shadows of Doubt
28999  986220                                RAN: Lost Islands

[1000 rows x 2 columns]
         appid                          name
29000   986280      Lingua Fleur: Lily
29001   986310                Recompile
29002   986320           AA Touch Gun!
29003   986340         Raid on the Ruhr
29004   986350       Schizm 3: Nemezis
...        ...                       ...
29995  1014310                Top Torch
29996  1014350            Once A Stray
29997  1014360  Adventures Diary of Merchant
29998  1014370  Bouncing DVD : The Game
29999  1014400                  Notemon

[1000 rows x 2 columns]
         appid                    name
30000  1014420           Blind Date
30001  1014450            Spacelair
30002  1014460            Nothing!
30003  1014470                 F8S
30004  1014480         Ecchi Girls
...        ...                    ...
30995  1041310        Dream Gallery
30996  1041320         Lords Mobile
30997  1041330  Tiger Fighter 1931 Tora!
30998  1041390      Legend of Cina
30999  1041400         Fight Angel
```

```
[1000 rows x 2 columns]
        appid                                          name
31000  1041450                                         RRRR2
31001  1041470                        三魂VR/The Spirits Within
31002  1041500                                  Dead Quest
31003  1041630                            Jungle Adventure
31004  1041650                           Time Warrior Z VR
...        ...                                           ...
31995  1066160  Captain Fly and Sexy Girls at the Night Club
31996  1066200                               The Brink 尘与土
31997  1066210                                Soviet Souls
31998  1066220                      Girls of Hentai Mosaic
31999  1066230                           Project Katharsis

[1000 rows x 2 columns]
        appid                                            name
32000  1066240                                          Tankex
32001  1066260                                Devolver Bootleg
32002  1066290                                      猎魔者战纪
32003  1066310                                        OFFSIDE
32004  1066370                                    Grand Heist
...        ...                                             ...
32995  1092710                           Hello Neighbor Alpha 1
32996  1092720                           Hello Neighbor Alpha 2
32997  1092730                           Hello Neighbor Alpha 3
32998  1092750                                       Boon Boon
32999  1092760  Story in the Dream World -Volcano And Possession-

[1000 rows x 2 columns]
        appid                                     name
33000  1092770                Hacking with Benefits
33001  1092780                             Depixtion
33002  1092800                Summit of the Wolf
33003  1092830          Coffee Runner Black and Mocha
33004  1092850                           Zeus Begins
...        ...                                   ...
33995  1120420                                Infini
33996  1120560  Sense - 不祥的预感: A Cyberpunk Ghost Story
33997  1120580                    Clans to Kingdoms
33998  1120600                      FromTheEarth VR
33999  1120640                    Kaiju Kite Attack

[1000 rows x 2 columns]
        appid                                          name
34000  1120680                                        ViVO
34001  1120690                                Knock Harder
34002  1120700                               Rescue Lucy 2
34003  1120770                Gods of Havoc: Fall to Earth
34004  1120780  Space Travelling within the Earth-Moon System
...        ...                                           ...
34995  1147880                                  Auto Fire
34996  1147900                                      炼妖记
34997  1147910                           Origami Ninja Star
34998  1147940                                       3dSen
34999  1147950                 Shio And Mysterious Forest

[1000 rows x 2 columns]
        appid                      name
35000  1147960                    Okaeri
35001  1147970                   Abode 2
35002  1147980       Gene Rain:Wind Tower
```

```
35003  1148000               Squirrel Jump
35004  1148010                   Love Chan
...        ...                         ...
35995  1177690               The Summoning
35996  1177710  Evolution Battle Simulator
35997  1177730                 W.H.A.L.E.
35998  1177810             Stream Fighters
35999  1177820               Shape Cascade

[1000 rows x 2 columns]
         appid                          name
36000  1177860                     Pink girl
36001  1177890             Cave of Illusions
36002  1177910                 Non-Compliant
36003  1177940                     Mass Plus
36004  1177950                Craft Elements
...        ...                           ...
36519  1201840                  MAN STANDING
36520  1201870              Drinks With Abbey
36521  1201910                   Cook Dungeon
36522  1202140  Portal Dungeon: Goblin Escape
36523  1202160                   Choco Pixel

[524 rows x 2 columns]
```

Initial Dataset Exploration

16 cells hidden

Preprocessing

The next step towards creating a regressor is to pre-process the data. As shown above, not all of the features are nu
requirement for a regressor or a classifier. As a team, we went through every feature and picked out ones that we th
on the review score. Furthermore, we specifically chose features that we thought could have an *explainable* impact
programmatic ways to do feature selection and elimination, due time constraints we opted for manual feature selec

The five "types" of transformations/preprocessing steps that we came up with are as follows:

1. Features that we need to indicate the presence of
2. Features that we need to count the number of
3. Features that we need to take the value of
4. Features that we need to convert into a one-hot vector
5. Features that we need to take summary stats of

In the first transformation, not all games have all features, but the fact that a feature exists, such as the `3rd-party_`
useful. This transformation should set the column to one if the feature exists, and zero otherwise.

In the second transformation, there exists certain features that are objects, or that have multiple values, such as the
contents aren't necessarily important, but the number of elements could be. This transformation should set the colu
elements that the feature contains. The default value should be zero.

In the third transformation, there exists features (boolean or numeric) that we just need to take the literal value of, su
This transformation should set the column to the normalized value of the feature (such as converting 'Yes' to 0). A d
chosen on a feature-by-feature basis.

In the fourth transformation, there exists categorical features that need to be converted into one-hot vectors and sp
columns so that they can be inputted into a regressor. This transformation should create columns for each categori
setting the specific column to one matching the specific element in any given row. This transformation should also t
the dataset, and as such filtering should be appropriately applied to ensure the list of categories is pruned.

In the fifth transformation, there exists time-series data, such as the `price_history` feature, that are not understooc
However, considering that these features are very useful, they should be represented in the model in some way. This
create multiple columns with summary statistics of the time series data.

Feature Presence Preprocessing

Double-click (or enter) to edit

```
def get_presence_of_df(chunk: pd.DataFrame) -> pd.DataFrame:
  #If the column does not exist in the chunk then we will just use a None value
  if('anti_cheat_support_url' not in chunk):
    chunk['anti_cheat_support_url'] = None
  if('party_account' not in chunk):
    chunk['party_account'] = None
  if('3rd-party_drm' not in chunk):
    chunk['3rd-party_drm'] = None
  if('demos' not in chunk):
    chunk['demos'] = None
  if('eulas' not in chunk):
    chunk['eulas'] = None

  df_presence_of = chunk[["3rd-party_account", "3rd-party_drm", 'anti_cheat_support_url','demos','eulas']]
  df_presence_of = df_presence_of.notnull().astype('int') #Converts NaN values to 0 and non NaN values to a

  return df_presence_of
```

Feature Count Preprocessing

```
def get_num_of(chunk: pd.DataFrame) -> pd.DataFrame:
  df_num_of = chunk[["achievements", 'dlcs','items', 'supported_languages', 'official_api_screenshots', 'st
  df_num_of["achievements"] = df_num_of["achievements"].str.len()
  df_num_of["dlcs"] = df_num_of["dlcs"].str.len()
  df_num_of["items"] = df_num_of["items"].str.len()
  df_num_of["supported_languages"] = df_num_of["supported_languages"].str.len()
  df_num_of["official_api_screenshots"] = df_num_of["official_api_screenshots"].str.len()
  df_num_of["stats"] = df_num_of["stats"].str.len()
  df_num_of.fillna(0, inplace=True)

  return df_num_of
```

Feature Value Preprocessing

```
value_of = ["allowpurchasefromrestrictedcountries", 'average_playtime_2_weeks','average_playtime_total',
            'cloud_quota', 'community_hub_visible', 'community_visible_stats', 'controller_support','contr
            'current_price','disablestreaming', 'dlcavailableonstore', 'exclude_from_game_sharing', 'exter
            'followers', 'hadthirdpartycdkey', 'has_adult_content', 'is_free', 'median_playtime_2_weeks',
            'median_playtime_total', 'onlyvrsupport','owners','player_count_all_time_peak','ram_min',
            'required_age', 'requireskbmouse', 'sourcegame', 'supports64bit', 'trading_card_drops', 'works

# ~ CORRECTION FUNCTIONS ~
def time_to_int(time):
    if type(time) is not str and np.isnan(time):
        #set default value
        num = 0
        t = "None"
    else:
        num, t = time.split()
        if t == "minutes":
            num = float(num) / 60
        else:
            num = locale.atof(num)
    return num

def convert_cloud_quota(quota):
    if type(quota) is not str and np.isnan(quota):
        #set default value
        return 0
    elif type(quota) is str and quota == 'empty':
        #set default value
        return 0
    else:
      num = quota.split()
      try:
          return num[2].strip('(').strip(')')
      except:
          return num[0].strip('(').strip(')')

def convert_current_price(price):
    try:
        return float(price.strip('$'))
    except:
        return 0

def yes_or_no_to_bool(toBool):
    toBool = str(toBool)
    if toBool == "No":
        return 0
    elif toBool == "Yes":
        return 1
    else:
        return 0

def controller_support_to_bool(controller):
    controller = str(controller)
    if controller == "partial":
        return 1
```

```
    elif controller == "full":
        return 1
    else:
        return 0

def float_to_int(x):
    try:
        return int(x)
    except:
        return 0

def value_of_vr_dict(x):
    if type(x) is not dict and np.isnan(x):
        #set default value
        return 0
    elif "{'kbm': 1, 'xinput': 1}" in str(x):
        return 1
    elif "{'steamvr' : 1}" in str(x):
        return 1
    else:
        return 0

def str_to_int(x):
    if type(x) is not str and np.isnan(x):
        #set default value
        return 0
    else:
        try:
            return int(x.replace(",",""))
        except:
            return x

def avg_owners(x):
    if type(x) is not str and np.isnan(x):
      #set default value
      x = 0
    else:
      split = x.split("..")
      sml = str_to_int(split[0])
      lrg = str_to_int(split[1])
      return int((lrg + sml) / 2)

    return x

def get_dict_values(x):
    if type(x) is not dict:
      return 0
    else:
      for item in x:
          try:
              val = locale.atoi(x[item])
          except:
              val = x[item]
          return val

# ~ MAIN FUNCTION ~

# Function for value_of
#   Takes in a data frame
```

```
#    Takes in a data frame
#    IF INT => leave as number
#    ELIF "yes" or "no" => convert to Boolean
#    ELSE => Fill in with value from dictionary of defualt values
#    Returns a data frame
def getValueOf(chunk: pd.DataFrame) -> pd.DataFrame:
    # Check for a columns existence; Initialize with none if not found in passed dataframe
    for col in value_of:
        if(col not in chunk):
            chunk[col] = 0

    # Select chunk of dataframe columns with given column names in value_of array
    df_value_of = chunk[value_of]

    # Loop over every attribute in value_of array
    for item in value_of:

        if item == 'community_hub_visible' or item == 'community_visible_stats' or item == 'has_adult_cont
            df_value_of[item] *= 1

        if item == "allowpurchasefromrestrictedcountries" or item == "externallyupdated" or item == 'hadth
            df_value_of[item] = df_value_of[item].apply(lambda x: yes_or_no_to_bool(x))

        if item == "controller_support":
            df_value_of[item] = df_value_of[item].apply(lambda x: controller_support_to_bool(x))

        if item == "dlcavailableonstore" or item == 'exclude_from_game_sharing' or item == 'required_age' 
            df_value_of[item] = df_value_of[item].apply(lambda x: float_to_int(x))

        if item == "controllervr":
            df_value_of[item] = df_value_of[item].apply(lambda x: value_of_vr_dict(x))

        # In Hours
        if item == "average_playtime_2_weeks" or item == "average_playtime_total" or item == "median_playt
            df_value_of[item] = df_value_of[item].apply(lambda x: time_to_int(x))

        # In MiB
        if item == "cloud_quota":
            df_value_of[item] = df_value_of[item].apply(lambda x: convert_cloud_quota(x))

        if item == "current_price":
            df_value_of[item] = df_value_of[item].apply(lambda x: convert_current_price(x))

        if item == "followers":
            df_value_of[item] = df_value_of[item].apply(lambda x: str_to_int(x))

        if item == "owners":
            df_value_of[item] = df_value_of[item].apply(lambda x: avg_owners(x))

        if item == "player_count_all_time_peak":
            df_value_of[item] = df_value_of[item].apply(lambda x: get_dict_values(x))

#         # <class 'numpy.int64'>
#         if item == 'disablestreaming':
#             df_value_of[item] = df_value_of[item].apply(lambda x: float_to_int(x))
#         # <class 'numpy.int64'>
#         if item == 'externalsubscriptiondlckey':
#             df_value_of[item] = df_value_of[item].apply(lambda x: float_to_int(x))
#         # <class 'numpy.int64'>
```

```
#         if item == 'onlyvrsupport':
#             df_value_of[item] = df_value_of[item].apply(lambda x: float_to_int(x))
#         # <class 'numpy.int64'>
#         if item == 'trading_card_drops':
#             df_value_of[item] = df_value_of[item].apply(lambda x: float_to_int(x))

    df_value_of.fillna(0, inplace=True)

    return df_value_of
```

Categorical Feature Preprocessing

```
# These are the columns we want:
category_list = ['Captions available',
 'Co-op',
 'Commentary available',
 'Cross-Platform Multiplayer',
 'Full controller support',
 'In-App Purchases',
 'Includes Source SDK',
 'Includes level editor',
 'LAN Co-op',
 'LAN PvP',
 'MMO',
 'Mods',
 'Mods (require HL2)',
 'Multi-player',
 'Online Co-op',
 'Online PvP',
 'Partial Controller Support',
 'PvP',
 'Remote Play Together',
 'Remote Play on Phone',
 'Remote Play on TV',
 'Remote Play on Tablet',
 'Shared/Split Screen',
 'Shared/Split Screen Co-op',
 'Shared/Split Screen PvP',
 'Single-player',
 'Stats',
 'Steam Achievements',
 'Steam Cloud',
 'Steam Leaderboards',
 'Steam Trading Cards',
 'Steam Turn Notifications',
 'Steam Workshop',
 'SteamVR Collectibles',
 'VR Support',
 'Valve Anti-Cheat enabled']

genre_list = ['Accounting',
 'Action',
 'Adventure',
 'Animation & Modeling',
 'Audio Production',
 'Casual',
```

```
 'Design & Illustration',
 'Early Access',
 'Education',
 'Free to Play',
 'Game Development',
 'Gore',
 'Indie',
 'Massively Multiplayer',
 'Movie',
 'Nudity',
 'Photo Editing',
 'RPG',
 'Racing',
 'Sexual Content',
 'Simulation',
 'Software Training',
 'Sports',
 'Strategy',
 'Utilities',
 'Video Production',
 'Violent',
 'Web Publishing']

primary_genre_list = ['0 (Unknown Genre)',
 '1 (Action)',
 '18 (Sports)',
 '2 (Strategy)',
 '21 (Unknown Genre)',
 '22 (Unknown Genre)',
 '23 (Indie)',
 '25 (Adventure)',
 '28 (Simulation)',
 '29 (Massively Multiplayer)',
 '3 (RPG)',
 '33 (Unknown Genre)',
 '34 (Unknown Genre)',
 '37 (Free to Play)',
 '4 (Casual)',
 '51 (Animation & Modeling)',
 '52 (Audio Production)',
 '53 (Design & Illustration)',
 '54 (Education)',
 '57 (Utilities)',
 '59 (Web Publishing)',
 '6 (Unknown Genre)',
 '70 (Early Access)',
 '71 (Sexual Content)',
 '72 (Nudity)',
 '73 (Violent)',
 '74 (Gore)',
 '9 (Racing)']

supported_system_list = ['Windows', 'Linux', 'macOS']

tag_list = ['1980s', "1990's",
 '2.5D', '2D', '2D Fighter', '360 Video',
 '3D', '3D Platformer', '3D Vision', '4 Player Local', '4X',
 '6DOF', 'ATV', 'Abstract', 'Action', 'Action RPG',
 'Action-Adventure', 'Addictive', 'Adventure', 'Agriculture', 'Aliens',
```

```
'Alternate History', 'America', 'Animation & Modeling', 'Anime', 'Arcade',
'Arena Shooter', 'Artificial Intelligence', 'Assassin', 'Asymmetric VR',
'Asynchronous Multiplayer', 'Atmospheric', 'Audio Production', 'Automation',
'BMX', 'Base Building', 'Baseball', 'Based On A Novel', 'Basketball',
'Batman', 'Battle Royale', "Beat 'em up", 'Beautiful', 'Benchmark',
'Bikes', 'Blood', 'Board Game', 'Bowling', 'Building', 'Bullet Hell',
'Bullet Time', 'CRPG', 'Capitalism', 'Card Game', 'Cartoon', 'Cartoony',
'Casual', 'Cats', 'Character Action Game', 'Character Customization',
'Chess', 'Choices Matter', 'Choose Your Own Adventure',
'Cinematic', 'City Builder', 'Class-Based', 'Classic',
'Clicker', 'Co-op', 'Co-op Campaign', 'Cold War', 'Colorful', 'Comedy',
'Comic Book', 'Competitive', 'Conspiracy', 'Controller', 'Conversation',
'Crafting', 'Crime', 'Crowdfunded', 'Cult Classic',
'Cute', 'Cyberpunk', 'Cycling', 'Dark', 'Dark Comedy', 'Dark Fantasy',
'Dark Humor', 'Dating Sim', 'Demons', 'Design & Illustration', 'Destruction',
'Detective', 'Difficult', 'Dinosaurs', 'Diplomacy', 'Documentary',
'Dog', 'Dragons', 'Drama', 'Driving', 'Dungeon Crawler', 'Dungeons & Dragons',
'Dynamic Narration', 'Dystopian', 'Early Access', 'Economy', 'Education',
'Emotional', 'Epic', 'Episodic', 'Experience', 'Experimental', 'Exploration',
'FMV', 'FPS', 'Faith', 'Family Friendly', 'Fantasy', 'Fast-Paced',
'Feature Film', 'Female Protagonist', 'Fighting', 'First-Person',
'Fishing', 'Flight', 'Football', 'Foreign', 'Free to Play', 'Funny',
'Futuristic', 'Gambling', 'Game Development', 'GameMaker', 'Games Workshop',
'Gaming', 'God Game', 'Golf', 'Gore', 'Gothic', 'Grand Strategy',
'Great Soundtrack', 'Grid-Based Movement', 'Gun Customization',
'Hack and Slash', 'Hacking', 'Hand-drawn', 'Hardware', 'Heist', 'Hex Grid',
'Hidden Object', 'Historical', 'Hockey', 'Horror', 'Horses', 'Hunting',
'Illuminati', 'Immersive Sim', 'Indie', 'Intentionally Awkward Controls',
'Interactive Fiction', 'Inventory Management', 'Investigation', 'Isometric',
'JRPG', 'Jet', 'Kickstarter', 'LEGO', 'LGBTQ+', 'Lara Croft', 'Lemmings',
'Level Editor', 'Linear', 'Local Co-Op', 'Local Multiplayer', 'Logic', 'Loot',
'Lore-Rich', 'Lovecraftian', 'MMORPG', 'MOBA', 'Magic', 'Management', 'Mars',
'Martial Arts', 'Massively Multiplayer', 'Masterpiece', 'Match 3', 'Mature',
'Mechs', 'Medieval', 'Memes', 'Metroidvania', 'Military', 'Mini Golf',
'Minigames', 'Minimalist', 'Mining', 'Mod', 'Moddable', 'Modern',
'Motocross', 'Motorbike', 'Mouse only', 'Movie', 'Multiplayer',
'Multiple Endings', 'Music', 'Music-Based Procedural Generation',
'Mystery', 'Mystery Dungeon', 'Mythology', 'NSFW', 'Narration',
'Nature', 'Naval', 'Ninja', 'Noir', 'Nonlinear', 'Nudity', 'Offroad',
'Old School', 'On-Rails Shooter', 'Online Co-Op', 'Open World', 'Otome',
'Parkour', 'Parody', 'Party-Based RPG', 'Perma Death', 'Philosophical',
'Photo Editing', 'Physics', 'Pinball', 'Pirates', 'Pixel Graphics',
'Platformer', 'Point & Click', 'Political', 'Politics', 'Pool',
'Post-apocalyptic', 'Procedural Generation', 'Programming', 'Psychedelic',
'Psychological', 'Psychological Horror', 'Puzzle', 'Puzzle-Platformer',
'PvE', 'PvP', 'Quick-Time Events', 'RPG', 'RPGMaker', 'RTS',
'Racing', 'Real Time Tactics', 'Real-Time', 'Real-Time with Pause',
'Realistic', 'Relaxing', 'Remake', 'Replay Value', 'Resource Management',
'Retro', 'Rhythm', 'Robots', 'Rogue-like', 'Rogue-lite',
'Romance', 'Rome', 'Runner', 'Sailing', 'Sandbox', 'Satire', 'Sci-fi',
'Science', 'Score Attack', 'Sequel', 'Sexual Content', "Shoot 'Em Up",
'Shooter', 'Short', 'Side Scroller', 'Silent Protagonist', 'Simulation',
'Singleplayer', 'Skateboarding', 'Skating', 'Skiing', 'Sniper',
'Snow', 'Snowboarding', 'Soccer', 'Social Deduction', 'Software',
'Software Training', 'Sokoban', 'Solitaire', 'Souls-like', 'Soundtrack',
'Space', 'Space Sim', 'Spectacle fighter', 'Spelling', 'Split Screen',
'Sports', 'Star Wars', 'Stealth', 'Steam Machine', 'Steampunk',
'Story Rich', 'Strategy', 'Strategy RPG', 'Stylized', 'Submarine',
```

```
 'Supernero', 'Supernatural', 'Surreal', 'Survival', 'Survival Horror',
 'Swordplay', 'Tactical', 'Tactical RPG', 'Tanks', 'Team-Based', 'Tennis',
 'Text-Based', 'Third Person', 'Third-Person Shooter', 'Thriller',
 'Time Attack', 'Time Management', 'Time Manipulation', 'Time Travel',
 'Top-Down', 'Top-Down Shooter', 'Touch-Friendly', 'Tower Defense',
 'TrackIR', 'Trading', 'Trading Card Game', 'Trains', 'Transhumanism',
 'Transportation', 'Turn-Based', 'Turn-Based Combat', 'Turn-Based Strategy',
 'Turn-Based Tactics', 'Tutorial', 'Twin Stick Shooter', 'Typing',
 'Underground', 'Underwater', 'Unforgiving', 'Utilities', 'VR', 'Vampire',
 'Video Production', 'Villain Protagonist', 'Violent', 'Visual Novel',
 'Voice Control', 'Voxel', 'Walking Simulator', 'War', 'Wargame',
 'Warhammer 40K', 'Web Publishing', 'Werewolves', 'Western', 'Word Game',
 'World War I', 'World War II', 'Wrestling', 'Zombies', 'eSports']

developer_25_list = ['Valve',
 'Elephant Games',
 'Telltale Games',
 'MumboJumbo',
 'Square Enix',
 'HeR Interactive',
 'SEGA',
 'Milestone S.r.l.',
 'Humongous Entertainment',
 'Arc System Works',
 'KOEI TECMO GAMES CO., LTD.',
 'Choice of Games',
 'Warfare Studios',
 'Creobit',
 'NedoStudio',
 'HexWar Games',
 'Hosted Games',
 'EnsenaSoft',
 'ERS G Studios',
 'Eipix Entertainment',
 'Tero Lunkka',
 'Ripknot Systems',
 'DRUNKEN APES',
 'Snkl Studio',
 'Laush Dmitriy Sergeevich',
 'Blender Games',
 'Nikita &amp;quot;Ghost_RUS&amp;quot;',
 'For Kids',
 'RewindApp',
 'CSM']

publisher_50_list = ['Strategy First',
 'Ubisoft',
 'Activision',
 'THQ Nordic',
 '1C Entertainment',
 '2K',
 'Focus Home Interactive',
 'SEGA',
 'Square Enix',
 'Big Fish Games',
 'Retroism',
 'Buka Entertainment',
 'Paradox Interactive',
 'Devolver Digital',
```

```
 'Microids',
 'Slitherine Ltd.',
 'Daedalic Entertainment',
 'BANDAI NAMCO Entertainment',
 'PLAYISM',
 'Degica',
 'United Independent Entertainment GmbH',
 'Plug In Digital',
 'Forever Entertainment S. A.',
 'MangaGamer',
 'KOEI TECMO GAMES CO., LTD.',
 'KISS ltd',
 'Artifex Mundi',
 'Sekai Project',
 'Choice of Games',
 'Alawar Premium',
 'Back To Basics Gaming',
 'Hosted Games',
 'Tero Lunkka',
 'Ripknot Systems',
 'Laush Studio',
 'Blender Games',
 'Ghost_RUS Games',
 'RewindApp']

# Here is the start of the function
def get_columns_that_exist(df, all_columns):
    """Get all column names from all_columns that exist in df and all that don't"""
    columns_that_exist = list()
    columns_that_dont = list()
    for col in all_columns:
        if col in df:
            columns_that_exist.append(col)
        else:
            columns_that_dont.append(col)
    return columns_that_exist, columns_that_dont


def list_join(row_list, col_prefix, val_wanted=None):
    """Join a list for get_dummies"""
    if type(row_list) is float:
        return ''
    if val_wanted is None:
        return '|'.join([col_prefix+x for x in row_list])
    else:
        return '|'.join([col_prefix+x[val_wanted] for x in row_list])


def one_hot_encode(df: pd.DataFrame) -> pd.DataFrame:
    """Returns a dataframe with the given features one-hot encoded"""
    df_one_hot = pd.DataFrame()

    features = ['categories', 'developer', 'genres', 'primary_genre', 'publisher', 'supported_systems', 't

    category_column_list = ['category_'+cat for cat in category_list]
    developer_column_list = ['developer_'+dev for dev in developer_25_list]
    genre_column_list = ['genre_'+genre for genre in genre_list]
    primary_genre_column_list = ['primarygenre_'+str(genre) for genre in primary_genre_list]
    publisher_column_list = ['publisher_'+pub for pub in publisher_50_list]
```

```
    publisher_column_list = ['publisher_'+pub for pub in publisher_50_list]
    supported_system_column_list = ['supportedsystem_'+system for system in supported_system_list]
    tag_column_list = ['tag_'+tag for tag in tag_list]

    for feature in features:
        if feature == 'categories':
            if feature not in df:
                dummies = pd.DataFrame(data=np.zeros((df.shape[0], len(category_column_list))), index=df.i
            else:
                category_descs = df['categories'].apply(lambda cat_list: list_join(cat_list, 'category_',
                dummies = category_descs.str.get_dummies(sep='|')
                columns_that_exist, columns_that_dont = get_columns_that_exist(dummies, category_column_li
                dummies = dummies[columns_that_exist]
                dummies_that_dont = pd.DataFrame(data=np.zeros((df.shape[0], len(columns_that_dont))), ind
                dummies = pd.concat([dummies, dummies_that_dont], axis=1)

        elif feature == 'developer':
            if feature not in df:
                dummies = pd.DataFrame(data=np.zeros((df.shape[0], len(developer_column_list))), index=df.
            else:
                dummies = pd.get_dummies(df['developer'], prefix='developer')
                columns_that_exist, columns_that_dont = get_columns_that_exist(dummies, developer_column_l
                dummies = dummies[columns_that_exist]
                dummies_that_dont = pd.DataFrame(data=np.zeros((df.shape[0], len(columns_that_dont))), ind
                dummies = pd.concat([dummies, dummies_that_dont], axis=1)

        elif feature == 'genres':
            if feature not in df:
                dummies = pd.DataFrame(data=np.zeros((df.shape[0], len(genre_column_list))), index=df.inde
            else:
                genres = df['genres'].apply(lambda genre_list: list_join(genre_list, 'genre_', 'descriptio
                dummies = genres.str.get_dummies(sep='|')
                columns_that_exist, columns_that_dont = get_columns_that_exist(dummies, genre_column_list)
                dummies = dummies[columns_that_exist]
                dummies_that_dont = pd.DataFrame(data=np.zeros((df.shape[0], len(columns_that_dont))), ind
                dummies = pd.concat([dummies, dummies_that_dont], axis=1)

        elif feature == 'primary_genre':
            if feature not in df:
                dummies = pd.DataFrame(data=np.zeros((df.shape[0], len(primary_genre_column_list))), index=
            else:
                dummies = pd.get_dummies(df['primary_genre'], prefix='primarygenre')
                columns_that_exist, columns_that_dont = get_columns_that_exist(dummies, primary_genre_colu
                dummies = dummies[columns_that_exist]
                dummies_that_dont = pd.DataFrame(data=np.zeros((df.shape[0], len(columns_that_dont))), ind
                dummies = pd.concat([dummies, dummies_that_dont], axis=1)

        elif feature == 'publisher':
            if feature not in df:
                dummies = pd.DataFrame(data=np.zeros((df.shape[0], len(publisher_column_list))), index=df.
            else:
                dummies = pd.get_dummies(df['publisher'], prefix='publisher')
                columns_that_exist, columns_that_dont = get_columns_that_exist(dummies, publisher_column_l
                dummies = dummies[columns_that_exist]
                dummies_that_dont = pd.DataFrame(data=np.zeros((df.shape[0], len(columns_that_dont))), ind
                dummies = pd.concat([dummies, dummies_that_dont], axis=1)

        elif feature == 'supported_systems':
            if feature not in df:
```

```
                dummies = pd.DataFrame(data=np.zeros((df.shape[0], len(supported_system_column_list))), in
            else:
                systems = df['supported_systems'].apply(lambda system_list: list_join(system_list, 'suppor
                dummies = systems.str.get_dummies(sep='|')
                columns_that_exist, columns_that_dont = get_columns_that_exist(dummies, supported_system_c
                dummies = dummies[columns_that_exist]
                dummies_that_dont = pd.DataFrame(data=np.zeros((df.shape[0], len(columns_that_dont))), ind
                dummies = pd.concat([dummies, dummies_that_dont], axis=1)

        elif feature == 'tags':
            if feature not in df:
                dummies = pd.DataFrame(data=np.zeros((df.shape[0], len(tag_column_list))), index=df.index,
            else:
                tags = df['tags'].apply(lambda tag_list: list_join(tag_list, 'tag_')) # '|'.join(['tag_'+t
                dummies = tags.str.get_dummies(sep='|')
                columns_that_exist, columns_that_dont = get_columns_that_exist(dummies, tag_column_list)
                dummies = dummies[columns_that_exist]
                dummies_that_dont = pd.DataFrame(data=np.zeros((df.shape[0], len(columns_that_dont))), ind
                dummies = pd.concat([dummies, dummies_that_dont], axis=1)

        df_one_hot = pd.concat([df_one_hot, dummies], axis=1)
    # TODO: Should this be returned/built as a DataFrame with sparse data?
    #       If we run into memory trouble, maybe. There are lots of zeros.
    return df_one_hot
```

Time Series Feature Preprocessing

```
summary_stats_of = ['playtime_and_viewer_count_all', 'price_history']

def __summary_statistics_helper(row: pd.Series) -> List[float]:
  if "playtime_and_viewer_count_all" in row:
    if type(row["playtime_and_viewer_count_all"]) == dict and "values" in row["playtime_and_viewer_count_a
      time_data = np.asarray(row["playtime_and_viewer_count_all"]["values"])
      time_data = time_data[time_data != np.array(None)]

      if len(time_data) > 0:
        playtime_stats = [np.min(time_data), np.max(time_data), np.mean(time_data), np.median(time_data)]
      else:
        playtime_stats = [0, 0, 0, 0]
    else:
      playtime_stats = [0, 0, 0, 0]

    if type(row["playtime_and_viewer_count_all"]) == dict and "values_twitch" in row["playtime_and_viewer_
      time_data = np.asarray(row["playtime_and_viewer_count_all"]["values_twitch"])
      time_data = time_data[time_data != np.array(None)]

      if len(time_data) > 0:
        twitch_stats = [np.min(time_data), np.max(time_data), np.mean(time_data), np.median(time_data)]
      else:
        twitch_stats = [0, 0, 0, 0]
    else:
      twitch_stats = [0, 0, 0, 0]
  else:
    playtime_stats = [0, 0, 0, 0]
    twitch_stats = [0, 0, 0, 0]
```

```
  if "price_history" in row:
    history_object = row["price_history"]
    if type(history_object) == dict:
      time_data = [float(history_object[price_time]["final"].replace("$", "")) for price_time in history_o
    else:
      time_data = []

    if len(time_data) > 0:
      price_stats = [np.min(time_data), np.max(time_data), np.mean(time_data), np.median(time_data)]
    else:
      price_stats = [0, 0, 0, 0]
  else:
    price_stats = [0, 0, 0, 0]


  return playtime_stats + twitch_stats + price_stats

def df_to_summary_statistics(df: pd.DataFrame) -> pd.DataFrame:
   df = df[summary_stats_of].apply(__summary_statistics_helper, axis=1, result_type="expand")
   df.columns = [
    "players_min",
    "players_max",
    "players_mean",
    "players_median",
    "twitch_viewers_min",
    "twitch_viewers_max",
    "twitch_viewers_mean",
    "twitch_viewers_median",
    "price_min",
    "price_max",
    "price_mean",
    "price_median",
   ]

   return df
```

Training Set Construction

With all of the preprocessing functions completed, the training set can be constructed by preprocessing each chunk results of the preprocessors.

```
dataset = pd.read_json("/content/drive/My Drive/Data Mining Shared Folder/database_new.jsonl.gz", orient="

df_combined_rows = pd.DataFrame()
df_labels = pd.DataFrame()

for chunk in dataset:
  # We only care about games
  chunk = chunk[chunk['app_type'] == 'Game']

  # Make sure we have a label
  chunk = chunk[chunk['rating_percent_positive'].notnull()]

  df1 = df_to_summary_statistics(chunk)
  df2 = get_num_of(chunk)
```

```
  df2 = get_num_of(chunk)
  df3 = get_presence_of_df(chunk)
  df4 = one_hot_encode(chunk)
  df5 = getValueOf(chunk)

  df_combined_columns = pd.concat([df1, df2, df3, df4, df5], axis=1)
  df_combined_rows = df_combined_rows.append(df_combined_columns, sort=True)
  df_labels = df_labels.append(chunk[["appid", "rating_percent_positive", "metacritic_score"]], sort=True)

df_combined_rows.to_csv('df_train.csv')
df_labels.to_csv('df_labels.csv')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  """
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing

/usr/local/lib/python3.6/dist-packages/pandas/core/frame.py:4259: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
  **kwargs
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:134: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:147: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:151: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:131: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:137: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:143: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:154: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:140: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:157: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:160: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing
```

Regressor Construction

With the training set constructed, the next step is to normalize it, and split it into training and test sets.

```
data_file = '/content/drive/My Drive/Data Mining Shared Folder/df_train.csv'
label_file = '/content/drive/My Drive/Data Mining Shared Folder/df_labels.csv'

label_col = 'rating_percent_positive'
id_col = 'appid'

# Load data
df = pd.read_csv(data_file, index_col=0, encoding="utf-8")
df_labels = pd.read_csv(label_file, index_col=0, encoding="utf-8")

## Preprocessing

# Normalize data (no longer a Pandas dataframe after this).
df = normalize(df, axis=0)

# Combine data with labels, to ensure every instance has the right label
# after the train/test split
```

```
# after the train/test split.
# Create np array big enough for data, appids, and labels.
combined_df = np.zeros((df.shape[0], df.shape[1]+2))

# Fill all but the last 2 columns with data.
combined_df[:, :-2] = df

# Put appids and labels as last 2 columns.
combined_df[:, -2:] = df_labels[[id_col, label_col]]

# Split out a test set and a validation set.
df_train, df_test = train_test_split(combined_df, test_size=0.1)
df_train, df_valid = train_test_split(df_train, test_size=0.1)

# Separate the data, appids, and labels again.
trainX = df_train[:, :-2]
trainY = df_train[:, -1]
trainIDs = df_train[:, -2]

testX = df_test[:, :-2]
testY = df_test[:, -1]
testIDs = df_test[:, -2]

validX = df_valid[:, :-2]
validY = df_valid[:, -1]
validIDs = df_valid[:, -2]

# Reduce the number of attributes/features;
# Resulting components explain at least 95% of the variance in the data.
pca = PCA(n_components=0.95, svd_solver='full')

# Find the PCA transform.
pca.fit(trainX)

# Transform the dataset via PCA.
trainX = pca.transform(trainX)

# Transform the test dataset via PCA.
testX = pca.transform(testX)

# Transform the validation dataset via PCA.
validX = pca.transform(validX)
```

```
##################
#    LassoCV    #
##################
from sklearn.feature_selection import VarianceThreshold
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LassoCV

data_file = '/content/drive/My Drive/Data Mining Shared Folder/df_train.csv'
df = pd.read_csv(data_file, index_col=0, encoding="utf-8")
sel = VarianceThreshold(threshold=(.80 * (1 - .80)))
sel.fit_transform(df)

## Preprocessing

# Normalize data (no longer a Pandas dataframe after this).
df = normalize(df, axis=0)
```

```
df = normalize(df, axis=0)

# Combine data with labels, to ensure every instance has the right label
# after the train/test split.
# Create np array big enough for data, appids, and labels.
combined_df = np.zeros((df.shape[0], df.shape[1]+2))

# Fill all but the last 2 columns with data.
combined_df[:, :-2] = df

# Put appids and labels as last 2 columns.
combined_df[:, -2:] = df_labels[[id_col, label_col]]

# Split out a test set and a validation set.
df_train, df_test = train_test_split(combined_df, test_size=0.1)
df_train, df_valid = train_test_split(df_train, test_size=0.1)

# Separate the data, appids, and labels again.
trainX = df_train[:, :-2]
trainY = df_train[:, -1]
trainIDs = df_train[:, -2]

testX = df_test[:, :-2]
testY = df_test[:, -1]
testIDs = df_test[:, -2]

validX = df_valid[:, :-2]
validY = df_valid[:, -1]
validIDs = df_valid[:, -2]

#Reduce the number of features using SelectFromModel
# We use the base estimator LassoCV since the L1 norm promotes sparsity of features.
clf = LassoCV()
threshold_value = 0.25
n_features_thresh = 465

#TrainX
# Set a minimum threshold of 0.25
sfm = SelectFromModel(clf, threshold=threshold_value)
sfm.fit(trainX, trainY)
n_features = sfm.transform(trainX).shape[1]

# Reset the threshold till the number of features equals threshold_value.
# Note that the attribute can be set directly instead of repeatedly
# fitting the metatransformer.
while n_features > n_features_thresh:
    sfm.threshold += 0.1
    trainX = sfm.transform(trainX)
    n_features = trainX.shape[1]

#TestX
sfm = SelectFromModel(clf, threshold=threshold_value)
sfm.fit(testX, testY)
n_features = sfm.transform(testX).shape[1]

while n_features > n_features_thresh:
    sfm.threshold += 0.1
    testX = sfm.transform(testX)
    n_features = testX.shape[1]
```

```
#ValidX
sfm = SelectFromModel(clf, threshold=threshold_value)
sfm.fit(validX, validY)
n_features = sfm.transform(validX).shape[1]

while n_features > n_features_thresh:
    sfm.threshold += 0.1
    validX = sfm.transform(validX)
    n_features = validX.shape[1]
```

Linear Regressor

While linear regressors are not inherently complex, they still provide good performance in some scenariors, and can for the future, more complex classifiers.

```
from sklearn.linear_model import LinearRegression

reg = LinearRegression().fit(trainX, trainY)

reg_score = reg.score(validX, validY)
```

```
print('linear regression:', reg_score)
```

```
linear regression: 0.2775647380692363
```

SVM

Support Vector Machine

```
from sklearn.svm import SVR

#svm_linear_regressor = SVR(kernel='linear').fit(trainX, trainY)
#svm_poly_regressor = SVR(kernel='poly').fit(trainX, trainY)
svm_rbf_regressor = SVR(kernel='rbf').fit(trainX, trainY)
#svm_sigmoid_regressor = SVR(kernel='sigmoid').fit(trainX, trainY)

#svm_linear_score = svm_linear_regressor.score(validX, validY)
#svm_poly_score = svm_poly_regressor.score(validX, validY)
svm_rbf_score = svm_rbf_regressor.score(validX, validY)
#svm_sigmoid_score = svm_sigmoid_regressor.score(validX, validY)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/base.py:193: FutureWarning: The default value of g
  "avoid this warning.", FutureWarning)
```

```
#print('linear:', svm_linear_score)
#print('polynomial:', svm_poly_score)
print('rbf:', svm_rbf_score)
#print('sigmoid:', svm_sigmoid_score)
```

```
rbf: -0.004185901242380163
```

Further Tuning

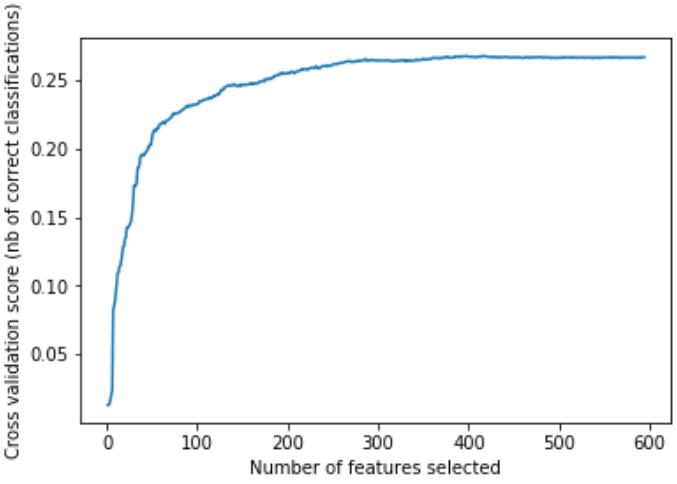
Feature Elimination

On their own, none of the regressors performed that well. The following sections seeks to fine tune and improve the regressors. The first potential tuning mechanism is programmatic feature elimimation. The following code fits a line recursively eliminates one feature at a time, preforming cross-validation every time a feature is eliminated. This allo optimal number of features. After figuring this out on the validation set, the model is re-trained on the full training se evaluated on the test set.

```
########################
#  Sam's Testing Code  #
########################
from sklearn.preprocessing import MaxAbsScaler
from sklearn.feature_selection import RFECV
from sklearn.svm import LinearSVR

data_file = '/content/drive/My Drive/Data Mining Shared Folder/df_train.csv'
label_file = '/content/drive/My Drive/Data Mining Shared Folder/df_labels.csv'

label_col = 'rating_percent_positive'
id_col = 'appid'

# Load data
df = pd.read_csv(data_file, index_col=0, encoding="utf-8")
df_labels = pd.read_csv(label_file, index_col=0, encoding="utf-8")

X_train, X_test, y_train, y_test = train_test_split(df, df_labels[[label_col]], train_size=0.75)

X_full_train = X_train
y_full_train = y_train

X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, train_size=0.80)

print(f"Number of training examples: {len(X_train)}")
print(f"Number of validation examples: {len(X_valid)}")
print(f"Number of full training examples: {len(X_full_train)}")
print(f"Number of test examples: {len(X_test)}")

svr = LinearSVR()

# Scale the training data
X_train_scaled = MaxAbsScaler().fit_transform(X_train)

rfecv = RFECV(estimator=svr, step=1, scoring='r2', cv=5, n_jobs=-1)
rfecv.fit(X_train_scaled, y_train.values.ravel())

print("Optimal number of features : %d" % rfecv.n_features_)

# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
```

```
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
```

```
Number of training examples: 18647
Number of validation examples: 4662
Number of full training examples: 23309
Number of test examples: 7770
Optimal number of features : 398
```




```
removed_features = []

for feature, mask in zip(X_train.columns.values, rfecv.support_):
  if mask == False:
    removed_features.append(feature)

print("The features removed with recursive feature elimination and cross-validation are:")
pprint.pprint(removed_features)

# Now we want to re-train the classifier on the full training set
svr_full = LinearSVR()

# Remove the un-important features
X_full_train_scaled = X_full_train[X_full_train.columns.difference(removed_features)]

# Apply the same scaling to the full training set
X_full_train_scaled = MaxAbsScaler().fit_transform(X_full_train_scaled)

# Train again
svr_full.fit(X_full_train_scaled, y_full_train.values.ravel())

# Scale/process the test sets as well
X_test_scaled = X_test[X_test.columns.difference(removed_features)]
X_test_scaled = MaxAbsScaler().fit_transform(X_test_scaled)

# Finally check our accuracy
print("Score on the test set after feature elimination:")
print(svr_full.score(X_test_scaled, y_test.values.ravel()))
```

```
The features removed with recursive feature elimination and cross-validation are:
['allowpurchasefromrestrictedcountries',
 'category_Captions available',
 'category_Co-op',
 'category_Full controller support',
 'category_Includes Source SDK',
 'category_LAN PvP',
 'category_Mods',
 'category_Mods (require HL2)',
 'category_Online Co-op',
 'category_Online PvP',
 'category_Remote Play on TV',
 'category_Shared/Split Screen',
 'category_Shared/Split Screen Co-op',
 'category_Stats',
 'category_Steam Workshop',
 'cloud_quota',
 'community_visible_stats',
 'developer_Choice of Games',
 'developer_Eipix Entertainment',
 'developer_KOEI TECMO GAMES CO., LTD.',
 'developer_Milestone S.r.l.',
 'disablestreaming',
 'eulas',
 'externallyupdated',
 'genre_Design & Illustration',
 'genre_Early Access',
 'genre_Movie',
 'genre_Nudity',
 'genre_RPG',
 'genre_Simulation',
 'genre_Software Training',
 'genre_Sports',
 'genre_Utilities',
 'genre_Video Production',
 'hadthirdpartycdkey',
 'has_adult_content',
 'items',
 'onlyvrsupport',
 'player_count_all_time_peak',
 'players_max',
 'primarygenre_0 (Unknown Genre)',
 'primarygenre_18 (Sports)',
 'primarygenre_22 (Unknown Genre)',
 'primarygenre_28 (Simulation)',
 'primarygenre_53 (Design & Illustration)',
 'publisher_2K',
 'publisher_Artifex Mundi',
 'publisher_BANDAI NAMCO Entertainment',
 'publisher_Choice of Games',
 'publisher_Daedalic Entertainment',
 'publisher_Forever Entertainment S. A.',
 'publisher_PLAYISM',
 'publisher_Paradox Interactive',
 'required_age',
 'stats',
 'supportedsystem_Linux',
 'supports64bit',
 'tag_1980s',
 'tag_2D',
 'tag_360 Video',
```

```
 'tag_3D',
 'tag_3D Vision',
 'tag_4 Player Local',
 'tag_ATV',
 'tag_Abstract',
 'tag_Action-Adventure',
 'tag_Adventure',
 'tag_Agriculture',
 'tag_Alternate History',
 'tag_Arcade',
 'tag_Arena Shooter',
 'tag_Artificial Intelligence',
 'tag_Assassin',
 'tag_Atmospheric',
 'tag_Base Building',
 'tag_Based On A Novel',
 'tag_Basketball',
 'tag_Battle Royale',
 'tag_Blood',
 'tag_Bowling',
 'tag_Building',
 'tag_Bullet Time',
 'tag_Capitalism',
 'tag_Cartoony',
 'tag_Character Action Game',
 'tag_Cinematic',
 'tag_Clicker',
 'tag_Co-op',
 'tag_Comic Book',
 'tag_Competitive',
 'tag_Crafting',
 'tag_Cycling',
 'tag_Dark',
 'tag_Dark Fantasy',
 'tag_Demons',
 'tag_Design & Illustration',
 'tag_Difficult',
 'tag_Dinosaurs',
 'tag_Dog',
 'tag_Drama',
 'tag_Driving',
 'tag_Dungeon Crawler',
 'tag_Early Access',
 'tag_Experience',
 'tag_FPS',
 'tag_Family Friendly',
 'tag_Fantasy',
 'tag_Feature Film',
 'tag_First-Person',
 'tag_Football',
 'tag_GameMaker',
 'tag_Grid-Based Movement',
 'tag_Hack and Slash',
 'tag_Hex Grid',
 'tag_Hidden Object',
 'tag_Hockey',
 'tag_Horror',
 'tag_Horses',
 'tag_Hunting',
 'tag_Indie',
 'tag_Intentionally Awkward Controls',
 'tag_Investigation',
```

```
        'tag_Isometric',
        'tag_Lara Croft',
        'tag_Local Co-Op',
        'tag_Local Multiplayer',
        'tag_Lore-Rich',
        'tag_MMORPG',
        'tag_Magic',
        'tag_Mature',
        'tag_Medieval',
        'tag_Metroidvania',
        'tag_Mini Golf',
        'tag_Motocross',
        'tag_Multiple Endings',
        'tag_Music',
        'tag_Mystery',
        'tag_NSFW',
        'tag_Narration',
        'tag_Noir',
        'tag_Nudity',
        'tag_Old School',
        'tag_On-Rails Shooter',
        'tag_Open World',
        'tag_Otome',
        'tag_Parody',
        'tag_Party-Based RPG',
        'tag_Physics',
        'tag_Post-apocalyptic',
        'tag_Puzzle-Platformer',
        'tag_PvE',
        'tag_RPG',
        'tag_Relaxing',
        'tag_Resource Management',
        'tag_Rogue-like',
        'tag_Rome',
        'tag_Sci-fi',
        'tag_Science',
        'tag_Sequel',
        'tag_Silent Protagonist',
        'tag_Skateboarding',
        'tag_Skating',
        'tag_Snow',
        'tag_Souls-like',
        'tag_Space',
        'tag_Spelling',
        'tag_Split Screen',
        'tag_Sports',
        'tag_Stealth',
        'tag_Strategy RPG',
        'tag_Supernatural',
        'tag_Surreal',
        'tag_Swordplay',
        'tag_Tactical',
        'tag_Tactical RPG',
        'tag_Tanks',
        'tag_Third Person',
        'tag_Thriller',
        'tag_Time Attack',
        'tag_Time Manipulation',
        'tag_Time Travel',
        'tag_Tower Defense',
        'tag_TrackIR',
        'tag_Transportation',
```

```
       'tag_Turn-Based',
       'tag_Turn-Based Strategy',
       'tag_Tutorial',
       'tag_Video Production',
       'tag_Villain Protagonist',
       'tag_War',
       'tag_Wargame',
       'tag_Warhammer 40K',
       'tag_Word Game',
       'tag_World War II',
       'trading_card_drops',
       'twitch_viewers_mean']
     Score on the test set after feature elimination:
     0.27461786605905136
```

Ensemble Regressors

While the performance did improve when utilizing recursive feature elimination, it did not improve all that much. Anc increase the regressor's score is to create an ensemble of classifiers, rather than using just one. Individual classifier specific subsets of the data, which allows the classifiers to fine-tune themselves to some specific domain of the dat account for a large number of features spanning different domains. From there, these classifiers can be combined i their predictions are averged (and weighted) to produce a final output.

While not shown below, for each of the ensemble regressors, they were constructed individually, and then a grid sea the hyperparameters on the validation set.

```
from sklearn.base import TransformerMixin, BaseEstimator
from sklearn.pipeline import Pipeline
from sklearn.ensemble import VotingRegressor, GradientBoostingRegressor, RandomForestRegressor
from sklearn.feature_selection import RFECV

# Adapted from https://ramhiser.com/post/2018-04-16-building-scikit-learn-pipeline-with-pandas-dataframe/
class ColumnSelector(BaseEstimator, TransformerMixin):
    def __init__(self, columns):
        self.columns = columns

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        assert isinstance(X, pd.DataFrame)
        return X[self.columns]


one_hot_pipe = Pipeline([
  ('extract', ColumnSelector(columns=[x for x in X_train.columns.values if x.startswith(('category_', 'tag_
  ('clf', RandomForestRegressor(n_estimators=100, n_jobs=-1, bootstrap=True, min_samples_leaf=8, min_sampl
])

time_series_pipe = Pipeline([
  ('extract', ColumnSelector(columns=["players_min", "players_max", "players_mean", "players_median", "twi
  ('clf',  GradientBoostingRegressor(min_samples_leaf=50, min_samples_split=10, n_estimators=300, subsampl
])

value_of_pipe = Pipeline([
```

```
value_of_pipe = Pipeline([
  ('extract', ColumnSelector(columns=[
            'allowpurchasefromrestrictedcountries', 'average_playtime_2_weeks','average_playtime_total',
            'cloud_quota', 'community_hub_visible', 'community_visible_stats', 'controller_support','contr
            'current_price','disablestreaming', 'dlcavailableonstore', 'exclude_from_game_sharing', 'exter
            'followers', 'hadthirdpartycdkey', 'has_adult_content', 'is_free', 'median_playtime_2_weeks',
            'median_playtime_total', 'onlyvrsupport','owners','player_count_all_time_peak','ram_min',
            'required_age', 'requireskbmouse', 'sourcegame', 'supports64bit', 'trading_card_drops', 'works
  ('clf', GradientBoostingRegressor(min_samples_leaf=30, min_samples_split=30, n_estimators=300, subsample=
])


# one_hot_pipe.fit(X_train, y_train.values.ravel())
# print("One hot forest pipe:")
# print(one_hot_pipe.score(X_valid, y_valid.values.ravel()))

# time_series_pipe.fit(X_train, y_train.values.ravel())
# print("Time series gard boost pipe:")
# print(time_series_pipe.score(X_valid, y_valid.values.ravel()))

# value_of_pipe.fit(X_train, y_train.values.ravel())
# print("Value of gard boost pipe:")
#print(value_of_pipe.score(X_valid, y_valid.values.ravel()))


ensemble_reg = VotingRegressor([('one_hot', one_hot_pipe), ('time_series', time_series_pipe), ('value_of',
ensemble_reg.fit(X_full_train, y_full_train.values.ravel())
print("Ensemble regressor score on the test set:")
print(ensemble_reg.score(X_test, y_test.values.ravel()))
```

```
Ensemble regressor score on the test set:
0.36945501659443886
```

Gradient Boosting Regressor

```
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingRegressor

grad = HistGradientBoostingRegressor(verbose=1, learning_rate=0.15)
grad.fit(X_full_train, y_full_trainy_test.values.ravel())
print("Gradient Boosting Regressor score on the test set:")
print(grad.score(X_test, y_test.values.ravel()))
```

```
Binning 0.111 GB of data: /usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:724: Dat
  y = column_or_1d(y, warn=True)
0.338 s
Fitting gradient boosted rounds:
[1/100] 1 tree, 31 leaves, max depth = 9, in 0.130s
[2/100] 1 tree, 31 leaves, max depth = 8, in 0.121s
[3/100] 1 tree, 31 leaves, max depth = 9, in 0.137s
[4/100] 1 tree, 31 leaves, max depth = 10, in 0.155s
[5/100] 1 tree, 31 leaves, max depth = 9, in 0.210s
[6/100] 1 tree, 31 leaves, max depth = 9, in 0.100s
[7/100] 1 tree, 31 leaves, max depth = 9, in 0.116s
[8/100] 1 tree, 31 leaves, max depth = 8, in 0.101s
[9/100] 1 tree, 31 leaves, max depth = 10, in 0.138s
[10/100] 1 tree, 31 leaves, max depth = 9, in 0.128s
[11/100] 1 tree, 31 leaves, max depth = 9, in 0.141s
[12/100] 1 tree, 31 leaves, max depth = 10, in 0.173s
[13/100] 1 tree, 31 leaves, max depth = 10, in 0.170s
[14/100] 1 tree, 31 leaves, max depth = 9, in 0.150s
[15/100] 1 tree, 31 leaves, max depth = 14, in 0.134s
[16/100] 1 tree, 31 leaves, max depth = 13, in 0.172s
[17/100] 1 tree, 31 leaves, max depth = 12, in 0.189s
[18/100] 1 tree, 31 leaves, max depth = 13, in 0.163s
[19/100] 1 tree, 31 leaves, max depth = 12, in 0.156s
[20/100] 1 tree, 31 leaves, max depth = 13, in 0.124s
[21/100] 1 tree, 31 leaves, max depth = 12, in 0.179s
[22/100] 1 tree, 31 leaves, max depth = 15, in 0.189s
[23/100] 1 tree, 31 leaves, max depth = 17, in 0.155s
[24/100] 1 tree, 31 leaves, max depth = 13, in 0.149s
[25/100] 1 tree, 31 leaves, max depth = 16, in 1.223s
[26/100] 1 tree, 31 leaves, max depth = 12, in 0.258s
[27/100] 1 tree, 31 leaves, max depth = 10, in 0.096s
[28/100] 1 tree, 31 leaves, max depth = 20, in 0.105s
[29/100] 1 tree, 31 leaves, max depth = 15, in 0.101s
[30/100] 1 tree, 31 leaves, max depth = 12, in 0.096s
[31/100] 1 tree, 31 leaves, max depth = 14, in 0.111s
[32/100] 1 tree, 31 leaves, max depth = 11, in 0.098s
[33/100] 1 tree, 31 leaves, max depth = 18, in 0.109s
[34/100] 1 tree, 31 leaves, max depth = 11, in 0.121s
[35/100] 1 tree, 31 leaves, max depth = 12, in 0.101s
[36/100] 1 tree, 31 leaves, max depth = 14, in 0.112s
[37/100] 1 tree, 31 leaves, max depth = 14, in 0.094s
[38/100] 1 tree, 31 leaves, max depth = 16, in 0.093s
[39/100] 1 tree, 31 leaves, max depth = 11, in 0.091s
[40/100] 1 tree, 31 leaves, max depth = 13, in 0.103s
[41/100] 1 tree, 31 leaves, max depth = 17, in 0.103s
[42/100] 1 tree, 31 leaves, max depth = 8, in 0.090s
[43/100] 1 tree, 31 leaves, max depth = 11, in 0.097s
[44/100] 1 tree, 31 leaves, max depth = 12, in 0.088s
[45/100] 1 tree, 31 leaves, max depth = 10, in 0.103s
[46/100] 1 tree, 31 leaves, max depth = 13, in 0.851s
[47/100] 1 tree, 31 leaves, max depth = 11, in 0.087s
[48/100] 1 tree, 31 leaves, max depth = 18, in 0.090s
[49/100] 1 tree, 31 leaves, max depth = 13, in 0.095s
[50/100] 1 tree, 31 leaves, max depth = 14, in 0.089s
[51/100] 1 tree, 31 leaves, max depth = 13, in 0.088s
[52/100] 1 tree, 31 leaves, max depth = 11, in 0.096s
[53/100] 1 tree, 31 leaves, max depth = 16, in 0.085s
[54/100] 1 tree, 31 leaves, max depth = 11, in 0.093s
[55/100] 1 tree, 31 leaves, max depth = 18, in 0.093s
[56/100] 1 tree, 31 leaves, max depth = 15, in 0.092s
[57/100] 1 tree, 31 leaves, max depth = 19, in 0.096s
```

```
[58/100] 1 tree, 31 leaves, max depth = 16, in 0.085s
[59/100] 1 tree, 31 leaves, max depth = 11, in 0.092s
[60/100] 1 tree, 31 leaves, max depth = 15, in 0.085s
[61/100] 1 tree, 31 leaves, max depth = 14, in 0.092s
[62/100] 1 tree, 31 leaves, max depth = 11, in 0.094s
[63/100] 1 tree, 31 leaves, max depth = 10, in 0.089s
[64/100] 1 tree, 31 leaves, max depth = 13, in 0.086s
[65/100] 1 tree, 31 leaves, max depth = 14, in 0.076s
[66/100] 1 tree, 31 leaves, max depth = 13, in 0.093s
[67/100] 1 tree, 31 leaves, max depth = 16, in 0.107s
[68/100] 1 tree, 31 leaves, max depth = 17, in 0.930s
[69/100] 1 tree, 31 leaves, max depth = 12, in 0.078s
[70/100] 1 tree, 31 leaves, max depth = 15, in 0.083s
[71/100] 1 tree, 31 leaves, max depth = 15, in 0.087s
[72/100] 1 tree, 31 leaves, max depth = 17, in 0.081s
[73/100] 1 tree, 31 leaves, max depth = 16, in 0.081s
[74/100] 1 tree, 31 leaves, max depth = 12, in 0.087s
[75/100] 1 tree, 31 leaves, max depth = 10, in 0.090s
[76/100] 1 tree, 31 leaves, max depth = 15, in 0.080s
[77/100] 1 tree, 31 leaves, max depth = 13, in 0.097s
[78/100] 1 tree, 31 leaves, max depth = 12, in 0.082s
[79/100] 1 tree, 31 leaves, max depth = 9, in 0.081s
[80/100] 1 tree, 31 leaves, max depth = 15, in 0.076s
[81/100] 1 tree, 31 leaves, max depth = 11, in 0.080s
[82/100] 1 tree, 31 leaves, max depth = 12, in 0.088s
[83/100] 1 tree, 31 leaves, max depth = 15, in 0.092s
[84/100] 1 tree, 31 leaves, max depth = 9, in 0.083s
[85/100] 1 tree, 31 leaves, max depth = 15, in 0.094s
[86/100] 1 tree, 31 leaves, max depth = 13, in 0.080s
[87/100] 1 tree, 31 leaves, max depth = 13, in 0.085s
[88/100] 1 tree, 31 leaves, max depth = 13, in 0.071s
[89/100] 1 tree, 31 leaves, max depth = 14, in 0.156s
[90/100] 1 tree, 31 leaves, max depth = 12, in 1.047s
[91/100] 1 tree, 31 leaves, max depth = 12, in 0.066s
[92/100] 1 tree, 31 leaves, max depth = 16, in 0.078s
[93/100] 1 tree, 31 leaves, max depth = 12, in 0.081s
[94/100] 1 tree, 31 leaves, max depth = 9, in 0.081s
[95/100] 1 tree, 31 leaves, max depth = 14, in 0.082s
[96/100] 1 tree, 31 leaves, max depth = 13, in 0.077s
[97/100] 1 tree, 31 leaves, max depth = 17, in 0.081s
[98/100] 1 tree, 31 leaves, max depth = 10, in 0.080s
[99/100] 1 tree, 31 leaves, max depth = 10, in 0.086s
[100/100] 1 tree, 31 leaves, max depth = 12, in 0.083s
Fit 100 trees in 14.762 s, (3100 total leaves)
Time spent computing histograms: 7.492s
Time spent finding best splits:  2.512s
Time spent applying splits:      2.272s
Time spent predicting:           0.013s
Gradient Boosting Regressor score on the test set:
0.4948917517418731
```

Here we can see, at least compared to the original classifiers, the performance has improved significantly.