Steam concurrent in-game users prediction based on concurrent users

Problem formulation:

In the past 20 years, Steam has exploded as a platform for people to buy, play and share their love for games. As a person that also love and enjoy playing games, I want to know how many of the concurrent users are playing in a game to see how active the user base is on a particular day.

Therefore, the datapoints in this project are days, each has different concurrent users count and concurrent in-game user count. The feature is the concurrent users. The label is the concurrent in-game users.

Summary:

- **Aim of project:** predict the concurrent in-game users on steam based on the concurrent users count
- **Data point:** a day
- **Feature(s):** Concurrent users on Steam (numerical – float)
- **Label(s):** Concurrent in-game users (numerical - float)

Method

Dataset

By sorting the data gathered from SteamDB [1] and Steam's own Steam & Game Stats page [2], I have created the current data set, which can be further updated as the data are gathered daily on the mentioned websites.

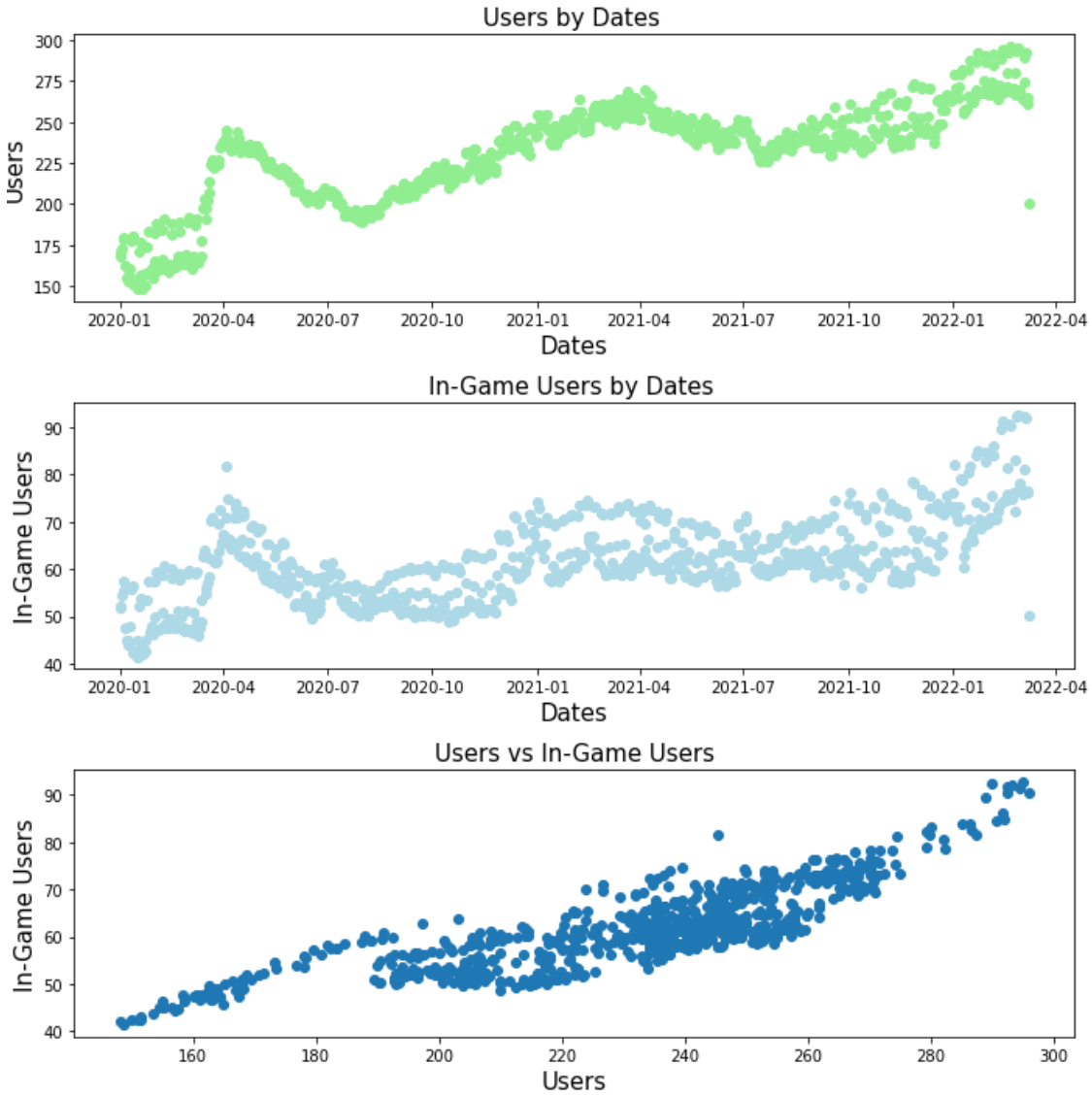
This dataset currently consists of 799 data points, each contains info of the date the data was gathered, the concurrent users and the concurrent in-game users of that day. The data set will be split into a test set (20%) and the remaining set (80%) for training and validating. This proportion enables enough data to be used for training and model selection. A test set has been created to evaluate overall performance and avoid overfitting to validation error. I will be using k-fold validation to produce more accurate errors estimation because it is applicable in this set, and is more efficient when compare with single split [3]

The data set consist of 3 column “Dates”, “Users” and “In-game” where 2 columns are possible candidates to act as feature.

Feature selection

The data points consist of the dates, the concurrent users, and concurrent in-game users. It easy to see the correlation between the two “users” column, if one is increasing, the other has a high chance of also increasing, while nothing can be gathered from the dates themselves.

By plotting the data points, the correlation between the two can be seen more clearly. Thus, the feature is the data from the “Users” column



(The numbers are in million)

Model selection

As seen in the plots, the concurrent users show a linear correlation with the concurrent in-game users. Hence, a linear method is suitable for this ML problem and thus, the linear regression method is chosen.
The label can be calculated using the formula:

$$\hat{y} = \alpha_1 x + \alpha_0$$

With y being the label (concurrent in-game users), x being the feature (concurrent users).

The mean squared error is calculated by the formula:

$$\text{MSE} = \frac{1}{k}\sum_{i=1}^{k}(y_i - \hat{y}_i)^2$$

Where k is the number of data points, y_i is the label and $\hat{y}_i$ the predicted label based on feature of the i^{th} data point. This loss function is chosen because it is often used in linear regression model [3], it is also supported by the sci-kit learn package with the function mean_squared_error which helps with the ease of use [4].

For k-fold validation, the dataset (remaining set) will be divided and folded with k = 7 (7 subsets, folded 7 times) to yield sufficient result without using too much time.

The Average results from the cross-validation iterations are:
+ Validation error: 18.225955735426755
+ Training error: 18.16471128494722

Reference List

[1] SteamDB: https://steamdb.info/app/753/graphs/

[2] Steam & Game Stats: https://store.steampowered.com/stats/Steam-Game-and-Player-Statistics

[3] Coursebook, Chapter 2.3

[4] Scikit Learn, https://scikit-learn.org/stable/index.html

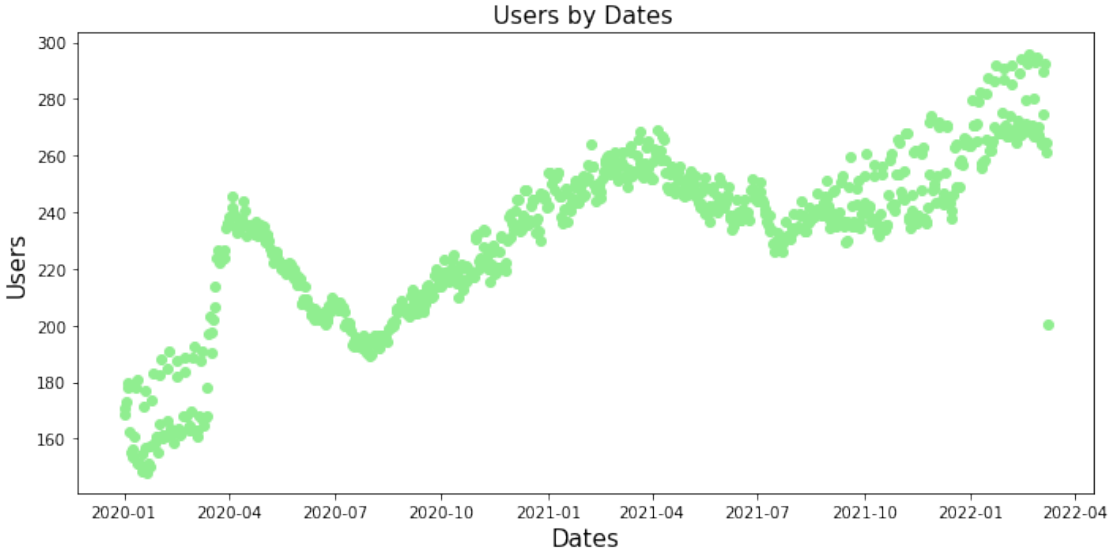
[5] Dataset Link: https://docs.google.com/spreadsheets/d/1OiPkJ-U3aKz5OXe9rUhGEUvOXE0WmUNTGHP8Nu9j4Dk/edit?usp=sharing

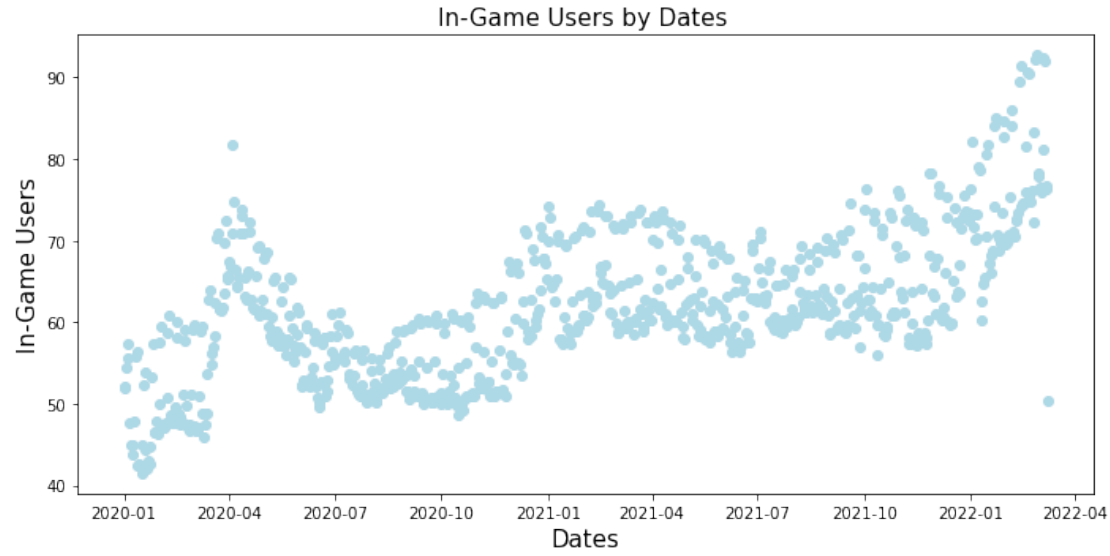
Appendix

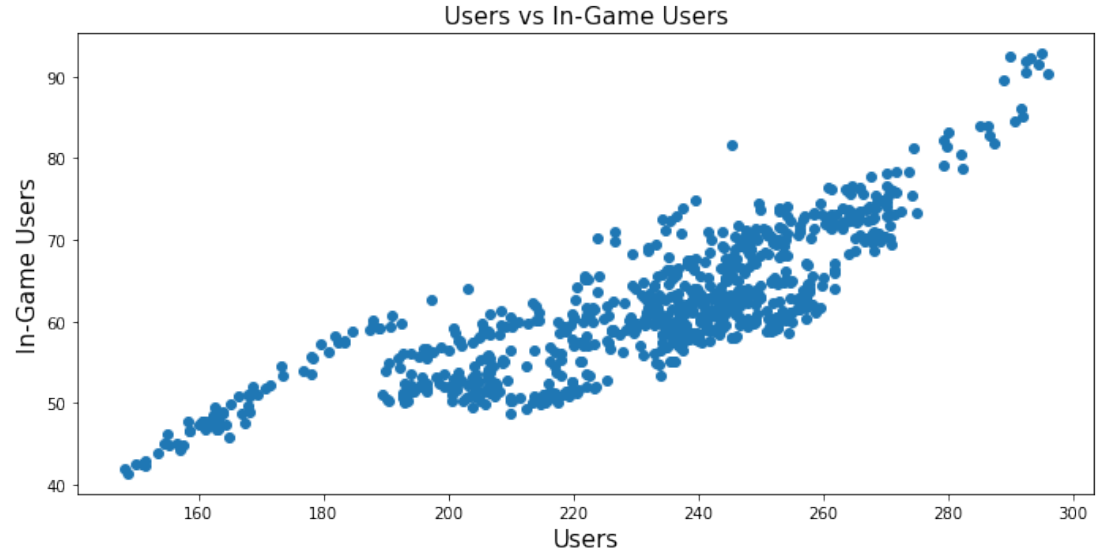
Data

March 9, 2022

[16]:
```
import pandas as pd
import matplotlib.pyplot as plt
SteamRaw = pd.read_excel("Steam.xlsx")
SteamRaw['Dates'] = pd.to_datetime(SteamRaw['Date']).dt.date
SteamRefined = SteamRaw.drop(columns=['Date'])
SteamRefined = SteamRefined[['Dates', 'Users', 'In-Game']]
SteamRefined['Users'] = SteamRefined['Users'].div(100000).round(2)
SteamRefined['In-Game'] = SteamRefined['In-Game'].div(100000).round(2)
SteamRefined.to_csv('Steam.csv')

fig, axes = plt.subplots(3, figsize=(10,15))
axes[0].scatter(SteamRefined['Dates'],SteamRefined['Users'],color =␣
↪'lightgreen')
axes[0].set_xlabel("Dates",size=15)
axes[0].set_ylabel("Users",size=15)
axes[0].set_title("Users by Dates ",size=15)

axes[1].scatter(SteamRefined['Dates'],SteamRefined['In-Game'],color =␣
↪'lightblue')
axes[1].set_xlabel("Dates",size=15)
axes[1].set_ylabel("In-Game Users",size=15)
axes[1].set_title("In-Game Users by Dates ",size=15)

axes[2].scatter(SteamRefined['Users'],SteamRefined['In-Game'])
axes[2].set_xlabel("Users",size=15)
axes[2].set_ylabel("In-Game Users",size=15)
axes[2].set_title("Users vs In-Game Users",size=15)
fig.tight_layout()
plt.show()
```


Users by Dates
Users
Dates
300
280
260
240
220
200
180
160
2020-01
2020-04
2020-07
2020-10
2021-01
2021-04
2021-07
2021-10
2022-01
2022-04


In-Game Users by Dates
In-Game Users
Dates
90
80
70
60
50
40
2020-01
2020-04
2020-07
2020-10
2021-01
2021-04
2021-07
2021-10
2022-01
2022-04


Users vs In-Game Users
In-Game Users
Users
90
80
70
60
50
40
160
180
200
220
240
260
280
300

[17]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

SteamCurrent = pd.read_csv('Steam.csv')
SteamCurrent = SteamCurrent[['Dates','Users','In-Game']]

tr_errors = []
val_errors = []
coefficients = []
intercepts = []
x = 0

X=SteamCurrent['Users'].to_numpy().reshape(-1,1)
y=SteamCurrent['In-Game'].to_numpy()
X_rem, X_test, y_rem, y_test = train_test_split(X,y,test_size=0.2,random_state␣
 ↪= 42)
k = KFold(n_splits=7,shuffle=True,random_state=42)


for (i, j) in k.split(X_rem):
    X_train, y_train, X_val, y_val = X[i], y[i],X[j], y[j]
    lin_regr = LinearRegression().fit(X_train, y_train)
    y_pred_train = lin_regr.predict(X_train)
    y_pred_val = lin_regr.predict(X_val)
    tr_error = mean_squared_error(y_train, y_pred_train)
    val_error = mean_squared_error(y_val, y_pred_val)
    val_errors.append(val_error)
    tr_errors.append(tr_error)
    intercepts.append(lin_regr.intercept_)
    coefficients.append(lin_regr.coef_)
    x += 1

    print("Iteration No.: " + str(x))
    print("Training error: " + str(tr_error))
    print("Validation error: " + str(val_error))
    print("\n")

print("Average val error: " + str(mean(val_errors)))
print("Average tr error: " + str(mean(tr_errors)))
print("\nLearned intercepts: " + str(intercepts))
print("Learned weights: "+ str(coefficients))
```

```
Iteration No.: 1
Training error: 18.1654204184952
Validation error: 18.21138340935131


Iteration No.: 2
Training error: 18.676897639571767
Validation error: 15.160294063340041


Iteration No.: 3
Training error: 17.767278609589095
Validation error: 20.67507390200928


Iteration No.: 4
Training error: 18.022163788314266
Validation error: 19.06303471517161


Iteration No.: 5
Training error: 18.179893648953584
Validation error: 18.103344149531516


Iteration No.: 6
Training error: 18.018447450903768
Validation error: 19.080424895976805


Iteration No.: 7
Training error: 18.32287743880288
Validation error: 17.288135012606705


Average val error: 18.22595573542675
Average tr error: 18.164711284947224

Learned intercepts: [13.352194809658066, 13.972285016232021, 14.325990867754733,
13.83281769292909, 13.674505028692721, 13.73113177678266, 13.866742648731282]
Learned weights: [array([0.20556819]), array([0.2027243]), array([0.20092714]),
array([0.20370758]), array([0.2041741]), array([0.20410542]),
array([0.20373828])]
```