Report

What are the main issues of **Steam** app users expericne in Play Market. Let's find out.

Descriptive analysis

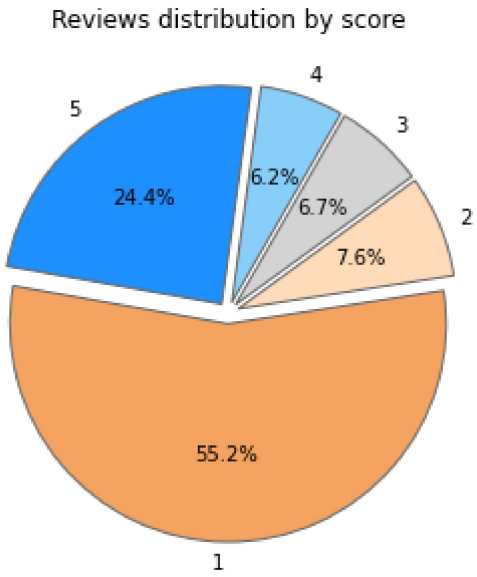
First, I have extracted newest 5000 customer reviews left in Play Market using **Google Play scrapper**.

Among extracted reviews the earliest dates to 15th April of 2021, while the latest update was at 1st June of 2021. So, we are safe, 5000 was enough to include all comments relevant to new version. I then deleted all outdated reviews (before 1st June)

After deleting, 4394 reviews left, mainly they have a score of 1, so the data was unballanced.

In [84]:
```
fig1
```

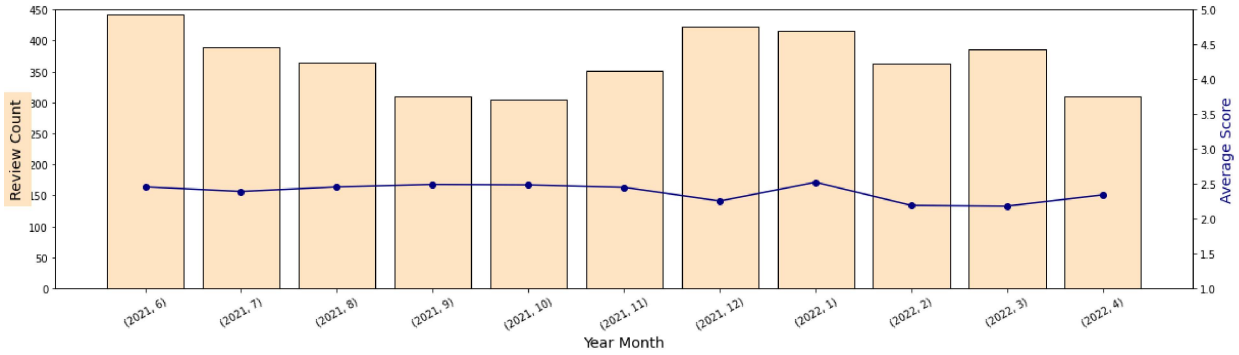
Out[84]:



Let's take a look at trends through the time. The number of reviews per month fluctuated between 300 and 450. While the average monthly score didn't show significant changes, and kept around 2.5. Nothing much insightful, but worth checking out.

In [85]:
```
fig2
```

Out[85]:

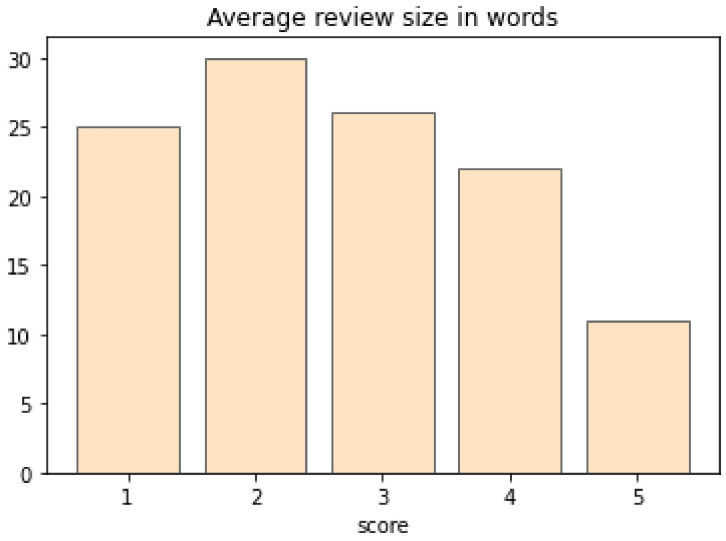


Reviews with score of **5** in average had significantly less (less than half) word count than each other score. That makes sense, as criticizing comments are more informative.

Although reviews of score **2** took only small portion (~8%) of all data, they had in average highest thumb-up count, 7 per review. Meaning score 2 reviews are mainly up to point.

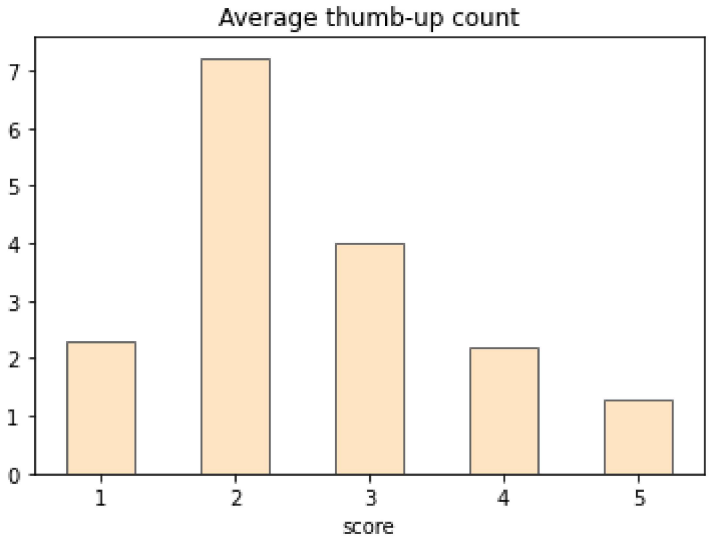
In [86]:
```
fig3
```

Out[86]:



In [87]:
```
fig4
```

Out[87]:



Strategy

The aim of this analysis is to obtain what are the main problems of the app. So we are going to classify bad and good reviews and see what drives this classification, that's the whole strategy. But here are the steps:

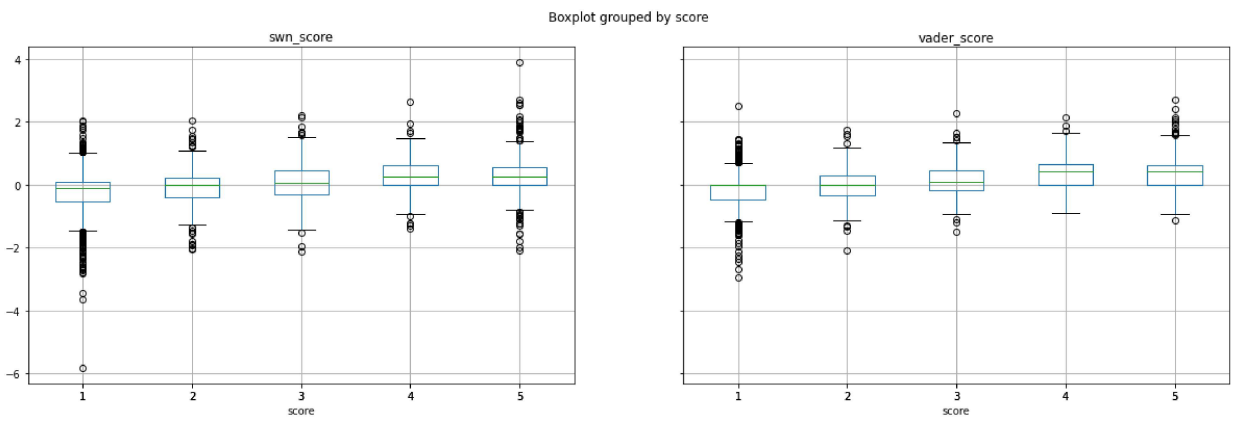
- First, we need to ensure that low scores really represent bad reviews, and accordingly high scores represent good reviews. This will be done by sentiment analysis.
- Next we need to label the reviews, i.e., bad or good, that's simply based on the scores that we validated in previous step
- Then we need to turn the words into features describing reviews
- Okay now we have features and labels, so the time to create ML models. It is crucial to use models than have an ability to extract feature importance. For this reason I have used Naive Bayes, Lasso Logistic Regression, Decision Tree, Random Forest, Ada Boost and linear Support Vector Machine.
- Then we are going to extract the feature importances (we secured before) from the best performing model. Reminding, that the features are just words, so we need to extract words that are apparently important for model and possibly insightful in context.
- Then we need to understand the meaning of selected features to gather insights.

Sentiment analysis

Sometimes users may leave scores inconsistent with the content of review. So, we need to check if the score really represented the sentiment of the review.

All right, for the sentiment analysis I used two algorithms: **SentWordNet** and **Vader**. Both algorithms indicate sentiment of each word in a review and then average them to one value. Basically they just represent how negative or positive review is based on the appearance of words they classify as negative or positive. Since algorithms also outputs numeric score, to not confuse you, I will say **sentiment score** to one caluclated by algorithms and **review score** to one left by customer in review.

In [89]:
```
raw_data.boxplot(column = ['swn_score', 'vader_score'], by = 'score', figsize = (20, 6
plt.show()
```



Fron the boxplot above we can see that with increase of sentiment score the review score increases as well, showing consistency. 2 rules can be emphasized from these boxplots, for both algorithms majority of:

1. Reviews with review score 1 and 2 had sentiment scores less than 0.5
2. Reviews with review score 4 and 5 had sentiment scores higher than 0

Still for these two rules above there were outliers. We are going to investigate them by looking at random sample. Here are random sample of reviews disobeying 1st rule:

In [134…
```
print(low_high_output)
```

```
Please i want to buy my game and i dont know the reason why my bank always decline, p
lease atleast let me know why my bank decline pleaseee

So recently I just got my account hacked and everything has been changed on it cuz wh
en I tried to reset my password I didn't get the email from it I have games on that a
ccount I would like my account back please.😔that's why I'm giving a one star but if
you could get my account back I would be happy to give it five stars. My username is
killergirlgamerwolf. I would very much appreciate if you could help

Every time I enter my email and password, it tells me that one of them is wrong, but
it won't tell me which. Much better on PC. This app needs improvement.

one star because i tryed to make a account but when i sayed that i am 16 years old no
thing happend so i hope that this will make it better

So thought I would try out steam on my new tablet, Now I have a question. Is there on
ly a special type of email that steam accepts??? I have tried all 4 of my emails incl
uding my business mail and according to your app none are valid. Just a thought, but
you may get more customers and thus make more profits if you fix your app.
```

Sample of reviews disobeying 2nd rule:

In [135…
```
print(high_low_output)
```

```
I cant wait to play ddlc

The people having problems are using potatoes.

Bad app.

The Dewey Decimal System What a scam that was

its ok but laggy and annoying sometimes i dunno
```

From these sample it is clear two issues. Firstly, people may leave really bad review but still put good score and vice-versa. Secondly, some reviews were just poorly classified by SWN and Vader algorigthms. While the latter issue just showing underperformance of algorithms, the former may cause some problems. Including bad reviews labelled by high score will poorly train

the model (garbage in, garbage out). So, for the sake of safety, reviews disobeying any of two rules were removed from the data.

In addition I found reviews with score 3 ambigous. By its nature it can be either good and bad evenly. So, to not confuse the model, reviews of score 3 were also removed.

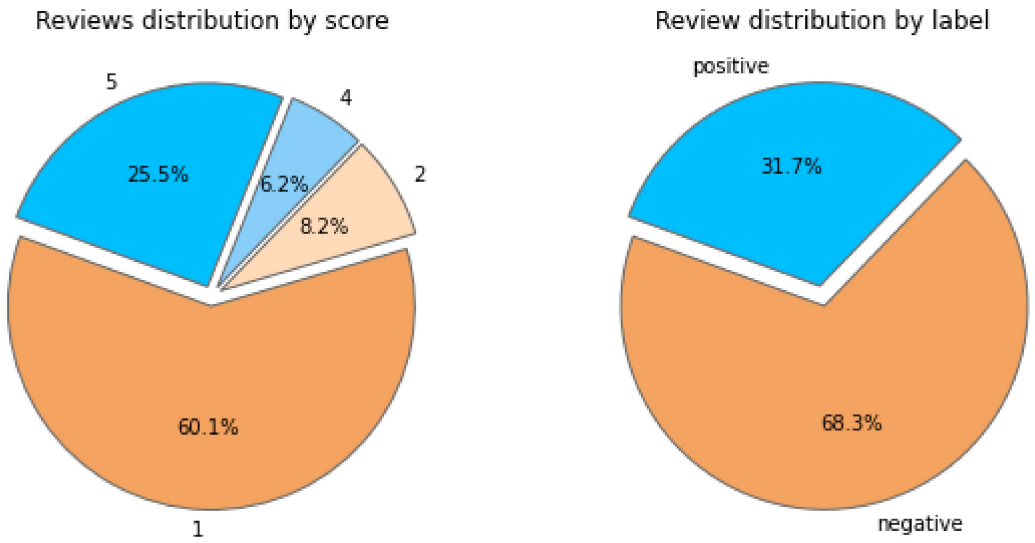
After removal of outliers and score-3 reviews we left with 3976 records out of initial 4394, which is around 90%. So, we sacrifised a small portion (10%) of data to ensure proper feed of models.

Labelling reviews

Now, after pervious step of sentiment analysis it is easy to label reviews. We don't have score 3. According to obtained rules review score of 1 and 2 had low to negative sentiemnt score, thefore can be labelled as negative. For the same reason review score of 4 and 5 are the positive reviews. We may see that after all the data is still unballanced, 2 negative reviews to 1 positive.

In [149…
```
fig6
```

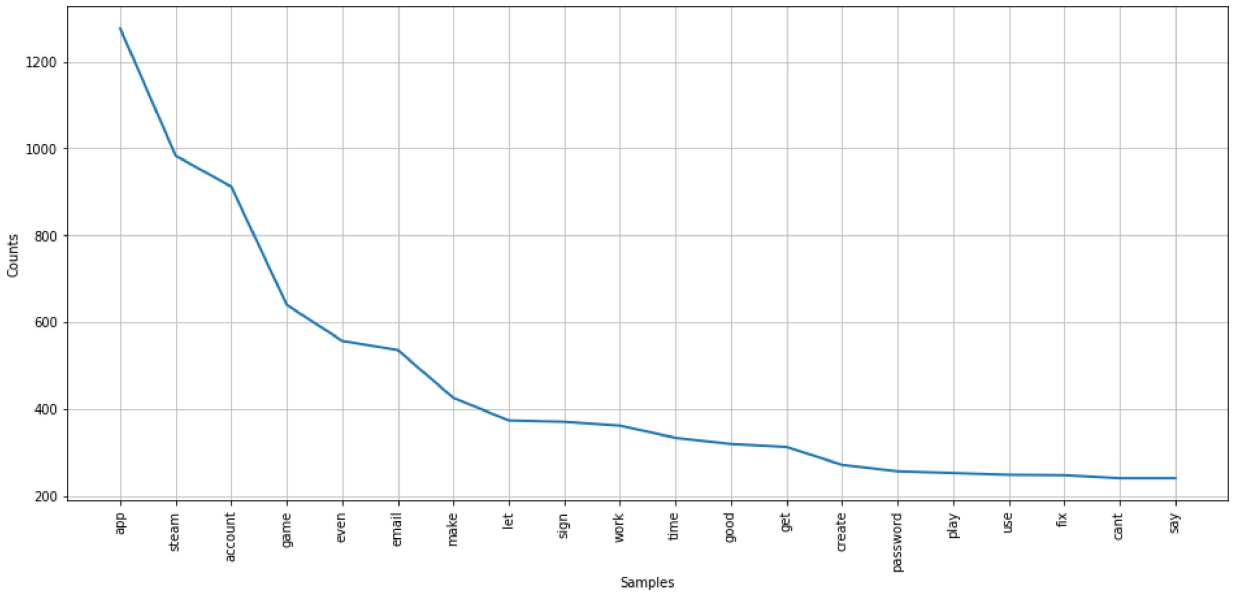
Out[149]:



Feature creation

From all 3976 reviews I have extracted the most frequently used words. The words were filtered from stop words and punctuations, so we left only with more or less meaningful words. As a rule of thumb I wanted to create a dataframe of dimensions approximately 1:20, so for 3970 reviews 200 features were needed. Therefore I have taken top 200 frequently used words for our analysis. Here is the first 20 of them and their frequency.

In [150…
```
popular_tokens = popular_strings_tokens(upd_data['content'], 200, True, 20)
```



We may see that some of words are not that meaningful (like word **let**) or obvious (like **app**), but I am going to keep it for two reasons. Fristly, we don't know what words have predictive force, let the models work on it. Secondly, some of the words may just work as a control variables, meaning they may not be the main predictor, but catches the changes in data, that weren't caught by the main predictor.

So, these 200 words were converted into feature of reviews in such a way: if a word (let's say **game**) appearce in a review its feature (**game**) would be 1, alternatively 0. So, we have only binary values.

ML models

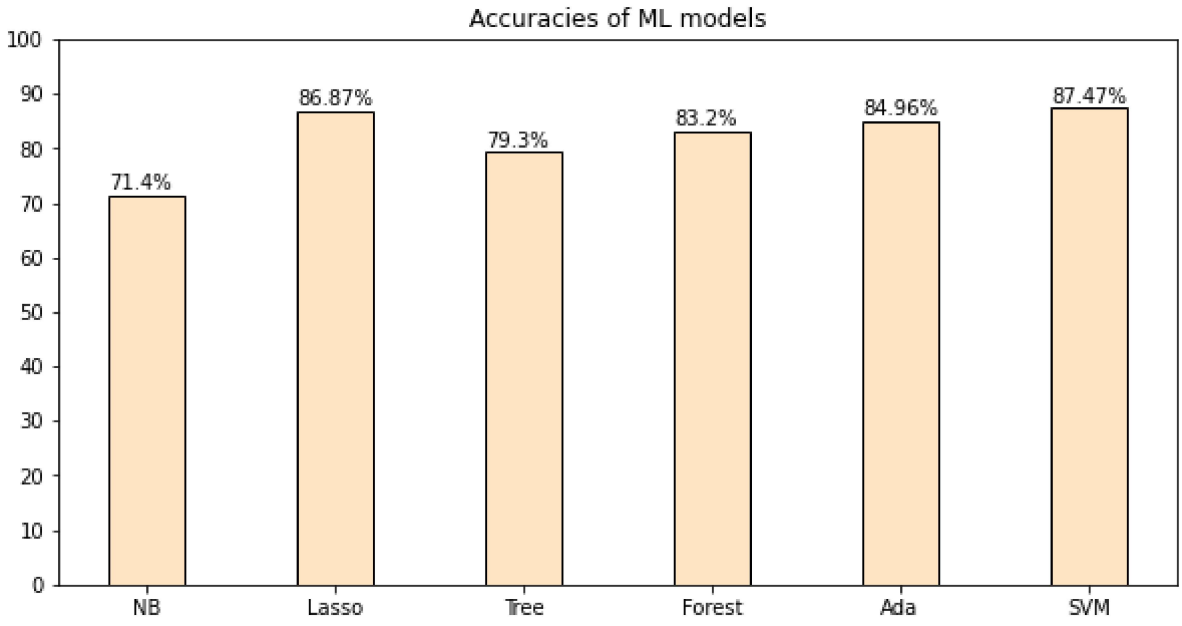
Previously I have mentioned that we are going to use 6 interpretable models. But, the tricky thing I am going to use them unusually.

Usually, we are training and testing model on different data, therefore making train-test split. Also, usually, we are making cross-validation during training. All this is done to avoid overfitiing the data, but we don't bother. We are not trying to extrapolate data, or predict score of future reviews, or preict score of other apps. We are trying to see what is going on with the specific app (Steam) at a specific period of time (since last update). Therefore we don't afraid of overfitting, but rather seeking it.

For the metric I used **balanced accuracy** to ensure models are trying to correctly predict both positive and negative reviews. As a result SVM achieved the best accuracy of 87.4%.

In [178…
```
fig8
```

Out[178]:



Feature importance

Since we used linear SVM it allows to extract coefficients of features it used for prediction. Since we are interested in features preicting negative reviews, we will look specifically at negative coefficients. Among them there were words like **horible** or **stupid**, their appearance is definitely a predictor of bad review, but they don't have much insights. On the other hand there were words like **install** or **captcha** with negative coefficient and potentially insightfull as well.

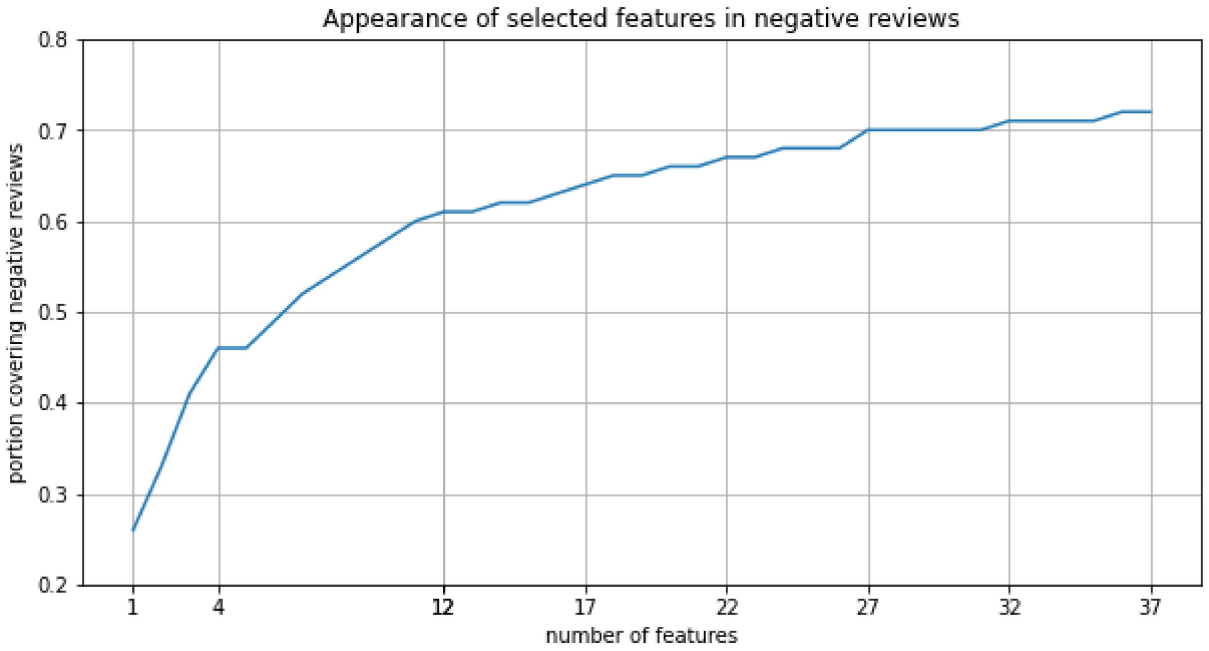
So, among all features with negative score, I have manually selected 37 of them that have a potential to reveal something meaningful. Overall they are covering more than 70% of all negative reviews, but not evenly.

As you may see below only first 4 features cover about 45% of all negative reviews, while first 12 features cover about 62%. Therefore it would not be wise to use all 37 features, when only 12 of them doing most of the work.

In [179…

```
fig9
```

Out[179]:

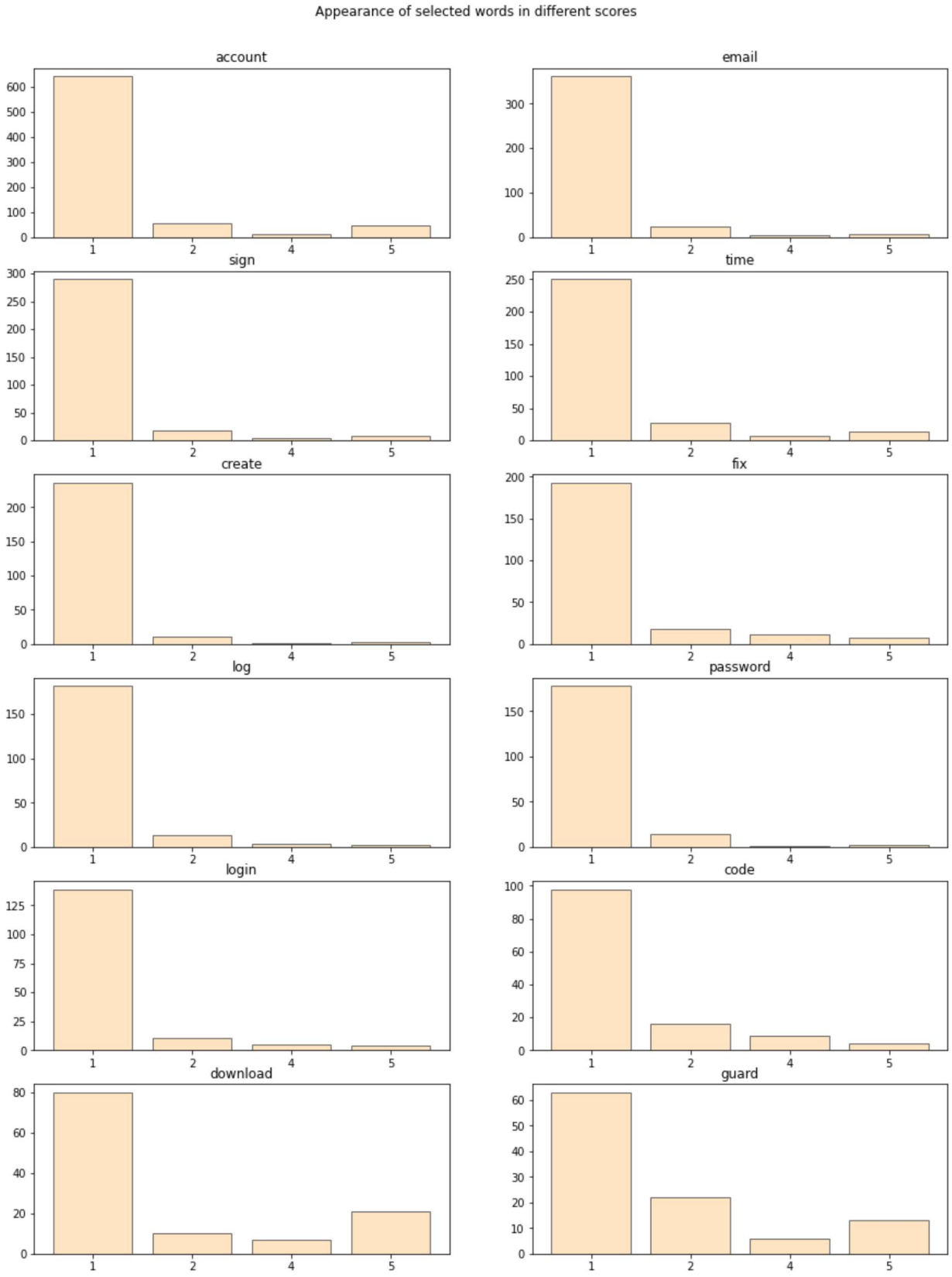


So, I have selected 12 features covering 62% of negative scores, what is next? We need to ensure that they are not covering majority of positive scores as well, otherwise they are just frequently appearing words, rather than predictors of bad reviews. Luckyly they are all distributed mainly in score 1 reviews

In [180…

```
fig10
```

Out[180]:



Context of selected features

The context of features were explored by directly looking at a sample of reviews contatining those features. As a result I revealed that word **time** is not that insightful, but used as a generic word like **every time** or **hundred times**. The word **fix** also appeared as a generic one, there

were nothing specifically mentioned to fix. And the word **download** were mainly used as a recommendation to not download this app.

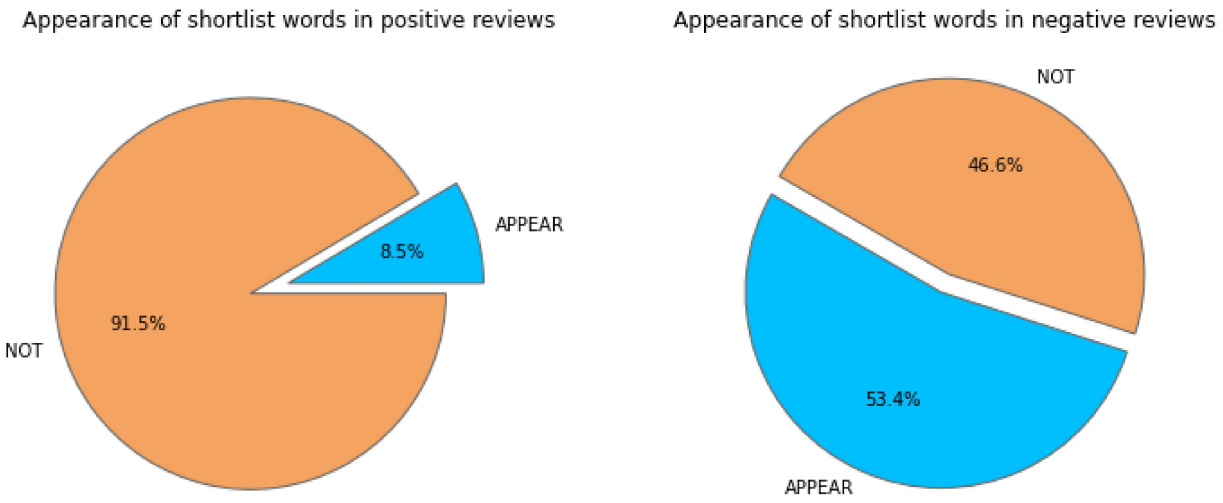
The rest 9 words were found to be more insightful and shared some few but common problems. Here is the main points of concern:

- It was difficult to create account: invalid email, infinite buffering time, restarts if minimize the app
- It was difficult to log in the app: wrong password, steam guard not sending code, lost phone = no access
- It was difficult to reset the password: invalid email, no email verificaiton

So, now we can see how these 9 words represent the main problems of **Steam** app. Now we need to check what portion of neagtive reviews are covered by only those 9 words, and how often these 9 words appear in positive reviews.

In [181…
```
fig11
```

Out[181]:



All right, now we see that there is insignificant appearance of selected words in positive reviews, while aggregatively they cover 53% of all negative reviews. From the whole analysis we can claim that half of negative reviews of mobile **Steam** app are related to **creation of account**, **log in to account** and **reseting the password**.

In [ ]:
```
```