Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Projektarbeit Seminar Statistical Learning

im Studiengang Wirtschaftsmathematik

Titel

Mobile Games: deskriptive Analyse und Vorhersageder durchschnittlichen Nutzerbewertung mithilfe Logistische Regression

Autor

Ngoc Mai Dinh

Matrikel-Nummer 568336

Betreuende Dozentin: Dr. Alla Petukhina

eingereicht am 27. Februar 2022

Inhaltsverzeichnis

1 Einleitung **3**

2 Methodik - Logistische Regression **5**
2.1 Definition 5
2.2 Logistische Regression versus Lineare Regression 5
2.3 Arten der logistischen Regression 6
2.4 Die Logistische Funktion 6
2.5 Schätzen des Regressionskoeffizienten 7
2.6 Modellbewertung 9

3 Empirische Analyse des Apple Apps Datasets **13**
3.1 Import und Datenbereinigung 13
3.2 Deskriptive Analyse und Visualisierung 14
3.3 Modellierung - Vorhersage der durchschnittlichen Nutzerbewertung . . 24

4 Fazit und Ausblick **28**

Literaturverzeichnis **30**

Abbildungsverzeichnis **31**

Tabellenvezeichnis **31**

1 Einleitung

Statistisches Lernen bezieht sich auf eine Vielzahl von Werkzeugen zum Modellieren und Verstehen komplexer Datensätze. Es ist ein neu entwickeltes Studiengebiet in der Statistik und vermischt sich mit parallelen Entwicklung in der Informatik und insbesondere dem maschinellen Lernen. Das Feld ist ein in den letzten Jahren viel diskutiertes Thema. Es ist ein Aspekt der künstlichen Intelligenz, der Verwendung in derzeit sehr unterschiedlichen Bereichen wie zum Beispiel der Finanzbranche, Robotik, Biologie, Landwirtschaft, Medizin und Astrophysik findet. Maschinelles Lernen ermöglicht Computern basierend auf Trainingsdaten selbstständig Vorhersagen oder Entscheidungen treffen zu können, ohne dass sie dafür spezielle programmiert wurden. Angewandt wird maschinelles Lernen oft, um entweder eine Vorhersage zu treffen (zum Beispiel die Prognose von Aktienkursen) oder um eine Klassifizierung vorzunehmen (Erkennen von Objekten). Darüber hinaus lässt sich maschinelles Lernen in die folgenden Teilbereiche einteilen:

- Überwachtes Lernen (Supervised Learning)
- Unüberwachtes Lernen (Unsupervised Learning)
- Bestärkendes Lernen (Reinforcement Learning).

Der Markt für mobile Handyspiele hat in den letzten Jahren deutlich zugenommen. Seit dem Jahr 2019 spielen mehr Deutsche Spiele auf ihrem Handy als auf der Konsole oder dem PC. Auch der Umsatz, den die Spieleunternehmen mit den Mobile Games erzielten, steigt stetig. Im Jahr 2021 lag dieser allein in Deutschland bei circa 2,3 Mrd. Euro. Aufgrund der Beliebtheit und des großen Marktes habe ich mir die Frage gestellt, welche Faktoren Einfluss auf den Erfolg und die Beliebtheit eines Spiels haben und wie man ein Mobile Game am besten im Markt positioniert.

Meine Projektarbeit kombiniert die beiden oben genannten Themengebiete. Ich beschäftige mich mit dem tieferen Verständnis einer Technik des maschinellen Lernens

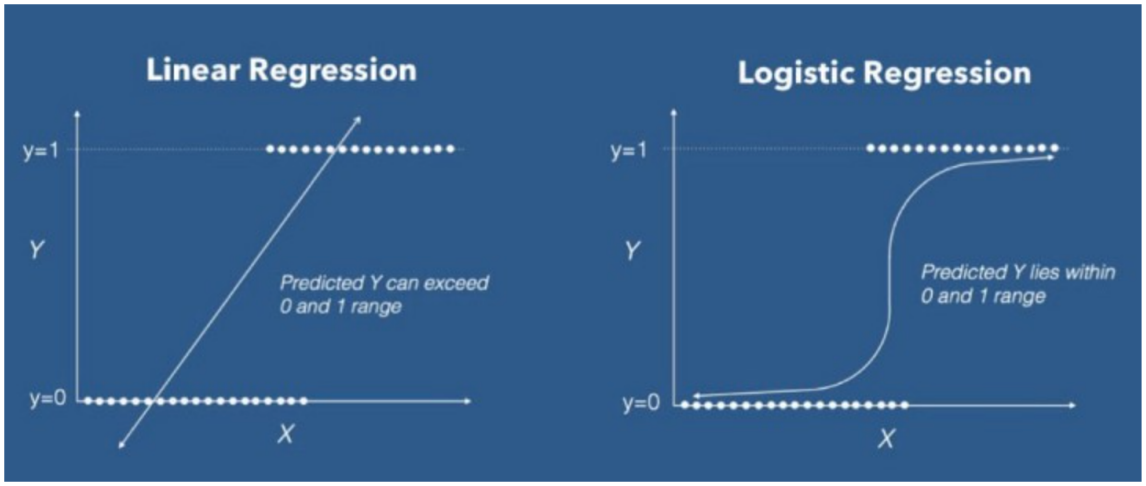
und zwar der Logistische Regression und baue darauf basierend ein Modell auf mit dem ich die durchschnittliche Nutzerbewertung eines Handyspiels vorhersagen kann. Diese Projektarbeit besteht aus zwei Hauptabschnitten. Der erste Abschnitt befasst sich mit den logistischen Regressionsmodellen. Während im Folgeabschnitt eine deskriptive Analyse und Visualisierungen des vorhanden Datensatzes zu den Mobile Games angefertigt wird und im Zuge dessen ein logistisches Regressionsmodell erstellt wird.

2 Methodik - Logistische Regression

2.1 Definition

Die logistische Regression modelliert die Wahrscheinlichkeiten für Klassifikationsprobleme mit zwei oder mehreren möglichen Ergebnissen. Es ist eine Erweiterung des linearen Regressionsmodells für Klassifizierungsprobleme. Ähnlich wie bei einer linearen Regressionsanalyse wird versucht, eine Funktionskurve zu finden, die möglichst gut zu den Daten passt. Diese Funktion ist jedoch im Gegensatz zur linearen Regressionsanalyse keine Gerade, sondern eine logistische Funktion mit einem s-förmigen Graphen.

2.2 Logistische Regression versus Lineare Regression



Bildquelle: Dev.to.

Abbildung 2.1: Linear Regression vs. Logistic Regression

Nicht nur bei der linearen sondern auch bei der logistischen Regression wird eine Prädiktorvariable verwendet, um eine Kriteriumsvariable vorherzusagen. Allerdings un-

terscheiden sich die beiden Formen der Regressionsanalyse in der Art ihres Kriteriums.

Bei der linearen Regression verwenden wir die abhängige Variable des Regressionsmodells, die mindestens intervallskaliert ist, was bedeutet dass wir die Werte des Kriteriums versuchen möglichst genau zu schätzen. Im Gegensatz dazu wird die logistische Regression benutzt, wenn die abhängige Variable nominalskaliert bzw. ordinalskaliert ist. Das heißt wir schätzen eine Wahrscheinlichkeit für eine der Kriteriumswerte, wobei der Wert zwischen 0 und 1 liegt.

2.3 Arten der logistischen Regression

Es gibt 3 Arten der logistischen Regression:

- Binäre logistische Regression: Die kategorische Antwort hat nur zwei mögliche Ergebnisse. Beispiel: Spam oder nicht.
- Multinomiale logistische Regression: Drei oder mehr Kategorien ohne Bestellung. Beispiel: Vorhersage, welches Essen bevorzugt wird (Vegetarisch, Nichtvegetarisch, Vegan).
- Ordinale logistische Regression: Drei oder mehr Kategorien wie bei der multiplen logistischen Regression, außer dass die kategorialen Zielvariablen geordnet sind. Beispiel: Rating von 1 bis 5.

2.4 Die Logistische Funktion

Die **logistische Regressionsfunktion** ist wie folgt definiert:

$$P(Y = 1|X = x) = \frac{e^z}{1+e^z}$$

und

$$P(Y = 0|X = x) = \frac{1}{1+e^z}$$

mit
$P(Y = 1|X = x)$: vorhergesagte Wahrscheinlichkeit, dass y = 1; wird auch als π bezeichnet
e : Basis des natürlichen Logarithmus, Eulersche Zahl
z : Logit (lineares Regressionsmodell der unabhängigen Variablen)
$\frac{e^z}{1+e^z}$: die Sigmoidfunktion

Die abhängige und unabhängige Variable bei der logistischen Regression hängen nicht linear zusammen. Das bedeutet, dass die Regressionskoeffizienten nicht auf die selbe Weise wie bei der linearen Regression interpretiert werden kann. Deshalb wird bei der logistischen Regression werden die so genannten **Odds** (oder **Wettquoten**) interpretiert:

$$odd(P(X)) = \frac{\pi}{1-\pi} = e^{\beta_0+\beta_1 x_1+\beta_2 x_2+...+\beta_k x_k+\epsilon}$$

$$0 \leq odd(\pi) \leq \infty$$

Der Quotient kann dabei beliebig viele positive Werte haben. Wird der Wert logarithmiert, sind unendliche Werte zwischen minus und plus möglich. Solche logarithmierten Odds werden in der Regel als "**Logit**"$\boldsymbol{z}$ bezeichnet.

$$z = ln(\frac{\pi}{1-\pi}) = \beta_0+\beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \epsilon$$

mit
x_k : die Eingangsvariable bzw. unabhängige Variable bezeichnet
β_0 : der Bias-Term
β_k: Regressionskoeffizienten bzw. das Gewicht/ Parameter für die Eingangsvariable x
ϵ : Fehlerwert

2.5 Schätzen des Regressionskoeffizienten

Wenn man die Methode der kleinsten Quadrate bei der linearen Regression benutzt, wird die **Maximum Likelihood Methode** angewandt, um die Modellparameter für die logistische Regressionsgleichung zu bestimmen.

Die Likelihood-Funktion

Für binäre y gilt $y_i \sim \mathrm{B}(1, \pi_i)$ mit $\pi_i = P(y_i = 1) = E(y_i) = \mu_i$ ist die Dichte durch

$$f(y_i|\pi_i) = f(y_i|\beta) = {\pi_i}^{y_i}(1 - \pi_i)^{1-y_i}$$

gegeben.

Die Likelihood L ist eine Funktion der unbekannten Parameter im Modell. Im Falle der logistischen Regression sind das $\beta_1, \beta_2, ..., \beta_k$. Daher $L(\beta)$ ist somit:

$$L(\beta) = \prod_{i=1}^{n} L_i(\beta) = \prod_{i=1}^{n} {\pi_i}^{y_i}(1 - \pi_i)^{1-y_i}$$

Maximum-Likelihood-Schätzer

Der Maximum Likelihood Schätzer kann bei der Schätzung von komplexen nichtlinearen wie auch linearen Modellen angewandt werden. Im Fall der logistischen Regression ist das Ziel, die Parameter β_k zu finden, die die sogenannte Log Likelihood Funktion $LL(\beta)$ maximieren. Die Log Likelihood Funktion ist der Logarithmus von $L(\beta)$:

$$log(L(\beta)) = \sum_{i=1}^{n} logL_i(\beta) = \sum_{i=1}^{n} (y_i log(\tfrac{\pi_i}{1-\pi_i}) + log(1 - \pi_i)) =$$

mit

$$\pi_i = P(y_i = 1) = E(y_i)$$

Für die Ableitung nach β bestimme zunächst:

Mit der Kettenregeln gilt

$$\frac{\partial logL(\beta)}{\partial \beta} = \tfrac{\partial logL_i}{\partial \eta_i} \tfrac{\partial \eta_i}{\partial \beta} = (y_i - \tfrac{e^{\eta_i}}{1+e^{\eta_i}})x_i$$

mit
$\eta = \beta_0 + \beta_1 x_i 1 + ... + \beta_k x_{ik} = \mathbf{x_i}^T \beta$
$\pi = \frac{e^\eta}{1+e^\eta}$ dann $\pi_i = \frac{e^{\mathbf{x_i}^T \beta}}{1+e^{\mathbf{x_i}^T \beta}}$

Die erste Ableitung der Log Likelihood funktion nach β ergibt sich die **Score Funktion**

$$s(\beta) = \frac{\partial log L(\beta)}{\partial \beta} = \sum_{i=1}^n \frac{\partial log L_i(\beta)}{\partial \beta} = \sum_{i=1}^n \mathbf{x_i}(y_i - \pi_i)$$

Die Bestimmung der ML-Schätzer β durch Lösen des nicht-linearen Gleichungssystem

$$s(\hat{\beta}) = \sum_{i=1}^n x_i(y_i - \frac{e^{\mathbf{x_i}^T \beta}}{1+e^{\mathbf{x_i}^T \beta}}) = 0$$

erfolgt nummerisch.

Numerische Bestimmung ML-Schätzer

- Newton-Verfahren, 1-dimensional

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

- Nullstelle der Score-Funktion, mehrdimensional

$$\hat{\beta}^{k+1} = \hat{\beta}^k + \frac{s(\hat{\beta}^k)}{H(\hat{\beta}^k)}$$

- Fischer-Scoring-Verfahren

$$\hat{\beta}^{k+1} = \hat{\beta}^k + \frac{s(\hat{\beta}k)}{F(\hat{\beta}^k)}$$

2.6 Modellbewertung

Es gibt zwei Hauptmetriken, um zu bewerten, wie gut ein Modell funktioniert, nachdem es trainiert wurde:

Konfusionsmatrix

Die Konfusionsmatrix ist eine Zusammenfassung von Vorhersageergebnissen zu einem Klassifizierungsproblem. Die Anzahl richtiger und falscher Vorhersagen wird mit Zählwerten zusammengefasst und nach Klassen aufgeschlüsselt. Die Konfusionsmatrix zeigt die Möglichkeiten unseres Klassifikationsmodells. Es zeigt uns nicht nur die Fehler auf, die von unserem Klassifikator gemacht werden, sondern vor allem die Arten von Fehlern, die gemacht werden. Es ist diese Aufschlüsselung, die die Einschränkung der alleinigen Verwendung der Klassifikationsgenauigkeit überwindet.

Um eine Konfusionsmatrix zu berechnen benötigen wir einen Testdatensatz mit erwarteten Ergebniswerten. Wir treffen eine Vorhersage für jede Zeile aus dem Testdatensatz. Von den erwarteten Ergebnissen und Vorhersagen zählen wir: Die Anzahl der richtigen Vorhersagen für jede Klasse und die Anzahl der falschen Vorhersagen für jede Klasse, organisiert nach der vorhergesagten Klasse.

Genauigkeit (eng. Accuracy)

Die Accuracy ist die am ehesten nachvollziehbare Metrik und repräsentiert den Prozentsatz der korrekt klassifizierten Proben.

$$Accuracy = \frac{\sum korrektklassifiziert}{\sum alleKlasszifierungen}$$

bzw.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

mit:

TP: true positive
TN: true negative
FP: false positive
FN: false negative

Receiver Operating Characteristic (ROC-Kurve)

ROC AUC stellt die Beziehung zwischen der wahren positiven Rate (TRP) - das heißt dem Verhältnis der Stichproben, die wir korrekt zur richtigen Klasse vorhergesagt haben - und der falsch positiven Rate (FPR)) - das heißt, das Verhältnis der Stichproben dar, für die wir ihre Klassenzugehörigkeit falsch vorhergesagt haben. Auf der y-Achse wird die Sensitivität aufgetragen, während auf der x-Achse 1-Spezifität aufgetragen wird, d.h. die Kurve dient der Diskrimination zwischen zwei Ereignissen.

Precision und Recall

Für Klassifikationsprobleme, bei denen die Datensätze der Klassen sehr unterschiedlich sind, wird häufig eine effektive Operation verwendet, nämlich Precision-Recall. Zunächst betrachten wir die binären Klassifikationsprobleme und zwar wenn eine der

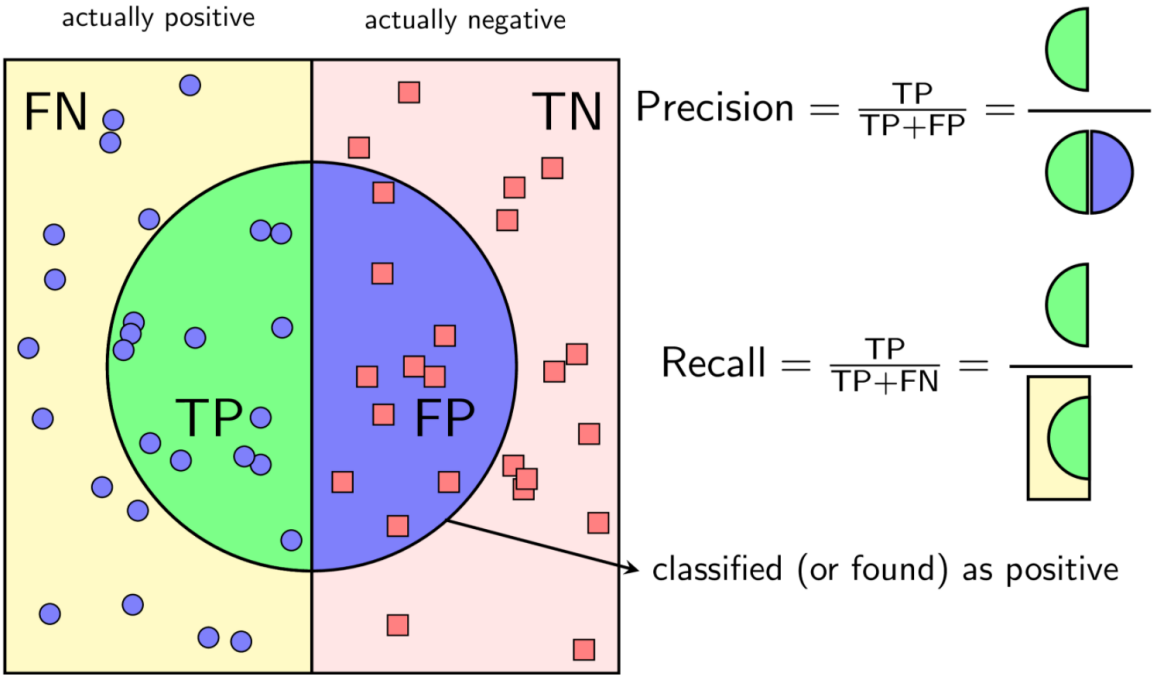


Abbildung 2.2: Precision und Recall

beiden Klassen als positiv, die andere als negativ bewertet wird. Die **Precision** ist definiert als das Verhältnis der richtig Positiven zu der Summe von richtig Positiven und als falsch positiv Klassifizierten **(TP + FP)**. **Recall** ist definiert als das Verhältnis

von richtig Positiven zu der Summe von richtig Positiven und falsch Negativen **(TP + FN)**.

$$Precision = \frac{TP}{TP+FP}$$

und

$$Recall = \frac{TP}{TP+FN}$$

mit $Precision \in [0; 1]$ und $Recall \in [0; 1]$

Hohe Precision bedeutet, dass die Genauigkeit der gefundenen Punkte hoch ist. Hoher Recall bedeutet, dass die Rate an fehlenden wirklich positiven Punkten gering ist. Ein gutes Klassifizierungsmodell ist, dass sowohl eine hohe Precision als auch einen hohen Recall hat, d.h. so nah wie möglich an eins ist. Es gibt zwei Möglichkeiten, die Qualität von Klassifikatoren basierend auf Precision und Recall zu messen: Precision-Recall-Kurve und F1-Score.

F1-Score

F1-Score ist das harmonische Mittel von Precision und Recall (angenommen dass sie beide ungleich Null sind):

$$\frac{2}{F1} = \frac{1}{Precision} + \frac{1}{Recall}$$

$$F1 = 2\frac{Precision.Recall}{Precision+Recall}$$

mit $F1 \in (0; 1]$

Je höher F1-Score, desto besser der Klassifikator. Am besten ist es, wenn sowohl Recall als auch Precision gleich 1 sind, F1 ist auch gleich 1.

3 Empirische Analyse des Apple Apps Datasets

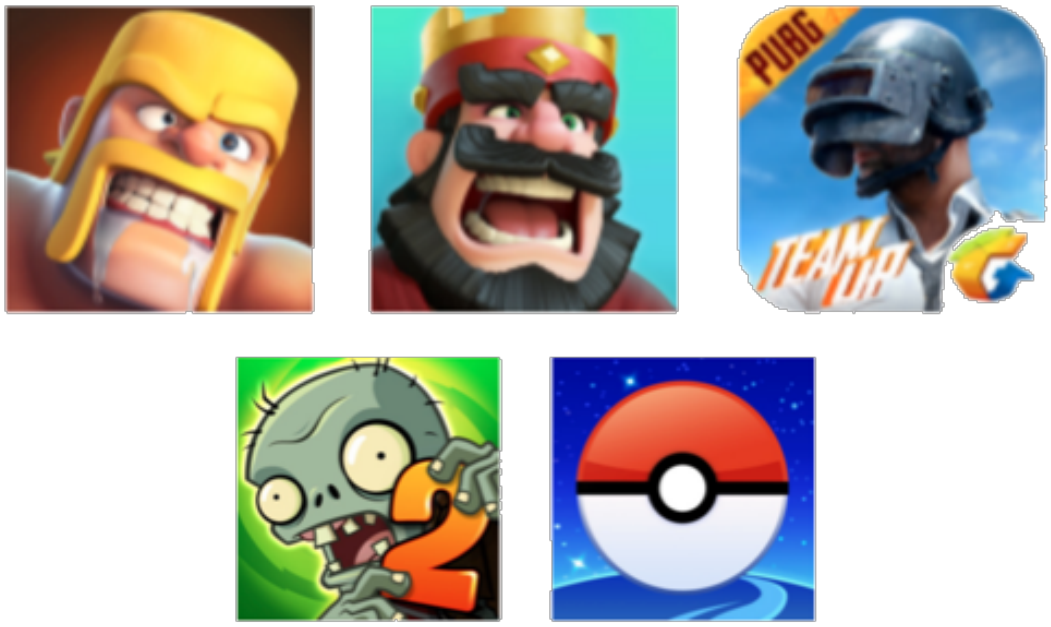


Abbildung 3.1: Top 5 Games der Datensatzes

3.1 Import und Datenbereinigung

Datenbereinigung ist einer der wichtigen Schritte vor der Analyse eines Datensatzes. Mithilfe der Python Bibliotheken *pandas, numpy, datetime* prüfe ich die Datensatzinformationen nach Features. Als nächstes möchte ich jegliche Duplikate im Datensatz entfernen und dann die Missing values behandeln. Danach benenne ich die Bezeichnungen der Spalten um und rechne die Gößе der Applikationen von Byte zu Megabyte um. Zum Schluss wird die Form des ursprünglichen Veröffentlichungsdatums und dem Veröffentlichungsdatum der aktuellen Version angepasst.

3.2 Deskriptive Analyse und Visualisierung

Zahlreichen Grafikbibliotheken wie *Matplotlib, Pandas Visualization, Seaborn, ggplot, Plotly* und weiteren Funktionen helfen uns dabei, die Daten effektiv zu visualisieren. Mit diesen Tools habe ich den Datensatz deskriptiv analysiert und visualisiert, um eine intuitivere Sicht auf die Datensätze zu bekommen.

Nutzerbewertung und ihre Anzahl

Zuerst betrachten wir die Nutzerbewertungen. Das Punktesystem ermöglicht eine Bewertung von 1 bis 5 Punkten in 0,5er Schritten. Wobei 1 die schlechteste und 5 die beste Bewertung ist. Somit können wir herausfinden, wie die Nutzer Gaming Apps am häufigsten bewerten.

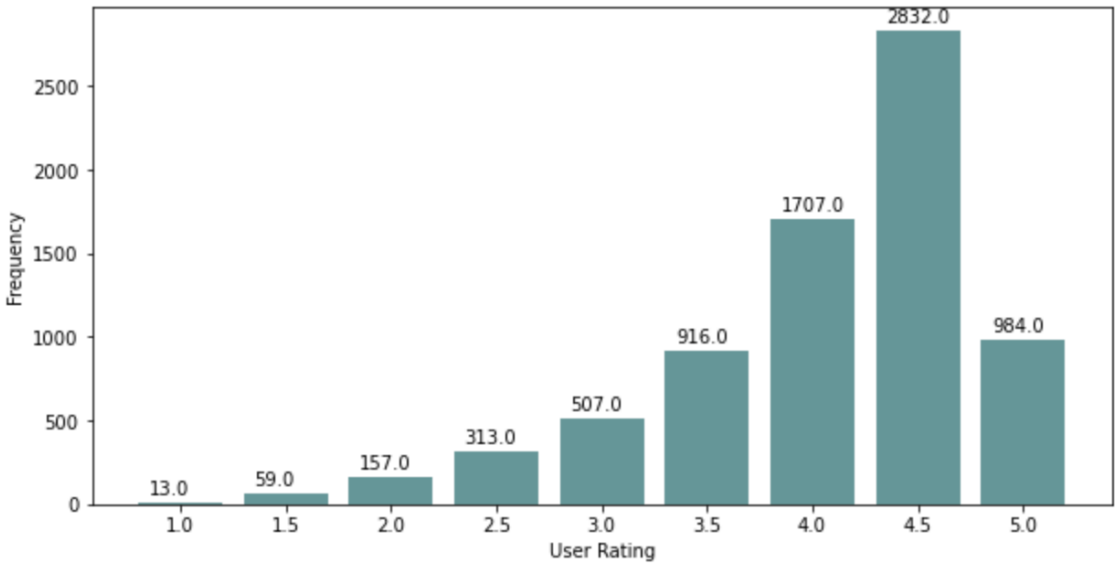


Abbildung 3.2: Nutzerbewertung vs. Anzahl der Nutzerbewertungen

Laut der Grafik werden die Spiele am häufigsten mit 4.5 Punkten bewertet. Die nächst häufigste Punktzahl ist 4.0 – jedoch mit einem großen Abstand zur 4,5. Außerdem wurden 5 Punkte seltener vergeben, als ich erwartet hatte. Allerdings scheinen die Spieler nicht wirklich streng zu sein, da die niedrigste Bewertung (1 Punkt), äußerst selten vorkommt. Lediglich 13 mal wurde eine App mit nur einem Punkt bewertet.

Anzahl der Games vs. Monetarisierungsmodell

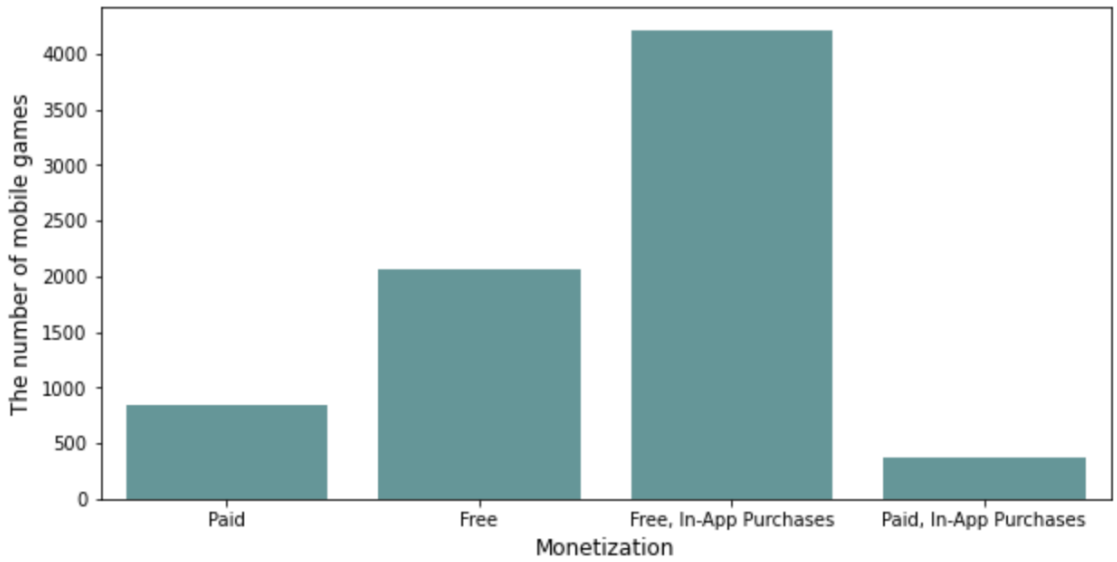


Abbildung 3.3: Anzahl der mobilen Games vs. Monetarisierung

Zunächst betrachten wir die Anzahl der Nutzerbewertungen nach Monetarisierungsmodell der Apps. Die Apps können wir hierbei in vier unterschiedliche Monetarisierungsmodelle unterteilen: Jene die zu 100% kostenlos sind, die die kostenlos sind aber In-App-Kauf Optionen beinhalten, andere die kostenpflichtig sind und die die kostenpflichtig und zudem noch In-App-Kauf Optionen beinhalten.

Wir sehen, dass die kostenlosen Spiele mit In-App Käufen die größte Gruppe sind, mit über 4000 Spiele. 100% kostenlosen Games sind die zweitgrößte Gruppe mit 2000 Spiele. Die kostenpflichtigen und zudem noch In-App-Kauf Optionen beinhaltende Spiele sind am seltensten. Dass die Benutzer gerne kostenlose Apps verwenden führt dazu, dass die Apps in den Suchergebnisse an die Spitze rücken, wodurch andere Benutzer auf diese aufmerksamer werden. Das hat zur Folge, dass sich der Effekt verstärkt und die kostenlosen Apps noch erfolgreicher werden.

Durchschnittliche Nutzerbewertung vs. Monetarisierungsmodell

Ähnlich wie soeben betrachten wir das Monetarisierungsmodell der Apps in Beziehung zu den Nutzerbewertungen. Der Grafik nach zu urteilen kann man sagen: Egal ob das Spiel In-App-Käufe hat, kostenpflichtig für den Download oder völlig kostenlos spielbar

ist, es scheint keinen Einfluss auf die Spielebewertung zu haben. Free-to-Play-Spiele haben lediglich einen sehr geringen Vorteil gegenüber den anderen Optionen.

Abbildung 3.4: Durchschnittliche Nutzerbewertung vs. Monetarisierung

Anzahl der Games nach Monetarisierungsmodell

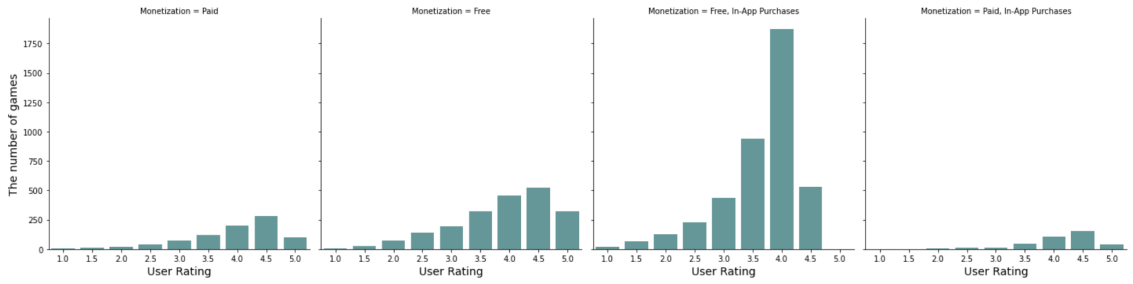


Abbildung 3.5: Anzahl der Games nach Monetarisierungsmodell

Hier sehen wir wie sich die Nutzerbewertungen nach den jeweiligen Monetarisierungsmodellen verhalten. Die Kurve von Free und Paid ohne In App Käufe ist sehr ähnlich. 4.0 und 4.5 Punkte wurden am häufigsten von Usern gegeben. Meistens gibt es fast genauso viele 4.0 Bewertungen wie 4.5 Bewertungen, lediglich bei kostenlosen Games mit In-App Käufen ist der Abstand sehr groß. Hier wurden am häufigsten 4.0 Punkte vergeben.

Nutzerbewertung vs. Alterbeschränkung

Eine weitere Frage ist, wie verhält sich die Beziehung zwischen den Nutzerbewertungen nach Altersbeschränkungen der Apps? Es gibt vier unterschiedliche Altersbeschränkungen: Ab 4+, 9+, 12+ und 17+ Jahren.

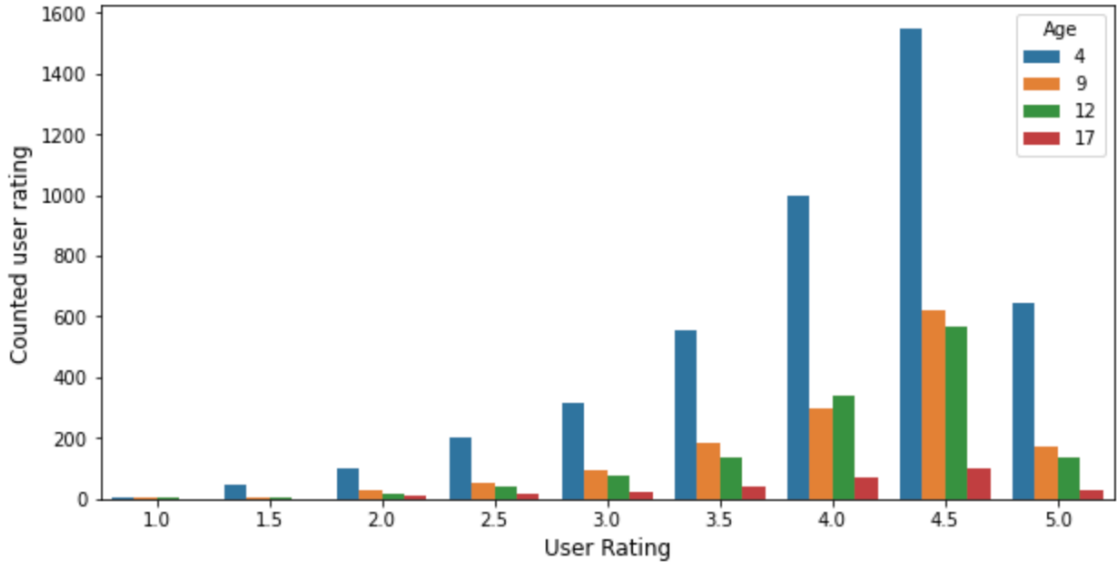


Abbildung 3.6: Benutzerbewertung vs. Alter

Die Altersstufen 9+ und 12+ sehen sehr ähnliche aus. Wobei bei allen Altersklassen zu erkennen ist, dass die Nutzer am häufigsten die Spiele mit 4,5 Punkten bewerten. Umso niedriger die Altersbegrenzung, desto mehr Bewertungen scheinen die Spiele zu bekommen. Das scheint logisch zu sein, denn das Spielehersteller Games produzieren wollen, die von möglichst vielen Zielgruppen gespielt werden können, ist wirtschaftlich sinnvoll.

Average User Rating, Size und Price

Im nächsten Schritt betrachten wir, wie die Beziehung zwischen der durchschnittlichen Nutzerbewertung, der Applikationsgröße in Megabyte und dem Monetarisierungsmodell ist.

Wie wir in der Grafik sehen können, hat das Monetarisierungsmodell kaum bis gar kei-

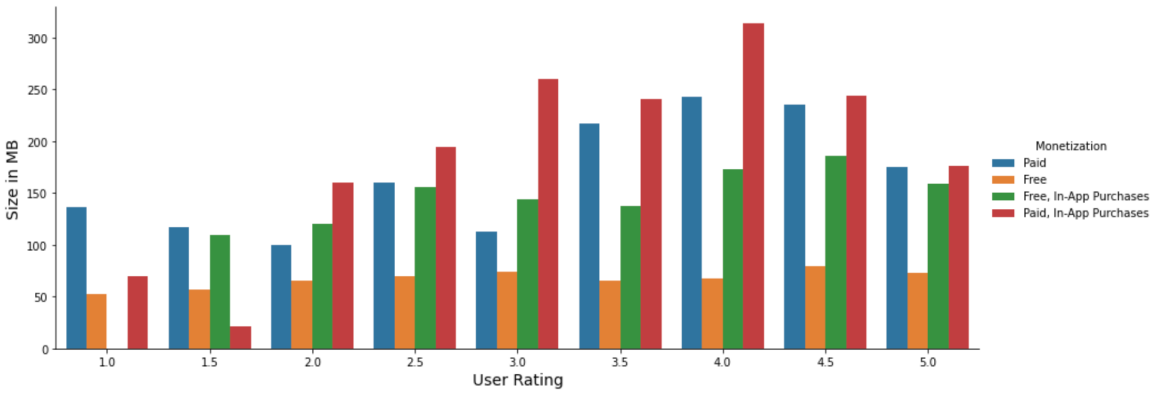


Abbildung 3.7: Beziehung zwischen Size, Preis und Nutzerbewertung

nen Einfluss auf die durchschnittliche Nutzerbewertung. Viele kostenpflichtige Spiele werden immer noch sehr hoch bewertet. Außerdem ist zu erkennen, dass kleine Applikationen häufiger kostenlos sind, als größere Spiele. Was zu erwarten war, da komplexere Spiele in der Regel auch aufwendiger für die Publisher zu entwickeln sind und dies unwirtschaftlich wäre, würden sie die Spiele völlig kostenlos zur Verfügung stellen.

Average User Rating vs. Primary Genre

Als nächstes betrachten wir, wie die Apps sortiert nach ihrem primären Genre von den Nutzern durchschnittlich bewertet werden. Es gibt insgesamt 21 primäre Genres, folgende Genre sind dabei am beliebtesten: Food & Drink (mit 5.0), News (4.75) und Music(4.5).

Dies sind jedoch Spielegenres mit einer sehr geringen Anzahl von Spielveröffentlichungen, nämlich ein Spiel in Food Drink, zwei Spiele in News und drei Spiele in Music. Das Genre "Games"hat die größte Anzahl von Veröffentlichungen (nämlich 7220 Spiele) und auch eine hohe Bewertung (über 4.0), weil dies ein beliebtes Genre ist, einfach zu spielen und für mehr Altersgruppen geeignet ist als andere Arten von Spielen.

Primary Genre	Menge der Games
Games	7220
Entertainment	92
Education	46
Utilities	44
Sport	23
Reference	18
Finance	8
Productivity	8
Book	5
Lifestyle	4
Music	3
Stickers	3
Social Networking	3
News	2
Business	2
Health and Finess	2
Travel	1
Shopping	1
Medical	1
Navigation	1
Food and Drink	1

Tabelle 3.1: Die Tabelle zeigt die Anzahl der mobilen Games nach Primary Genre. Siehe das in Python-Skript.

Sprache des Spiels

Als nächstes möchte ich visualisieren, welche Sprachen die Spiele am häufigsten unterstützen.

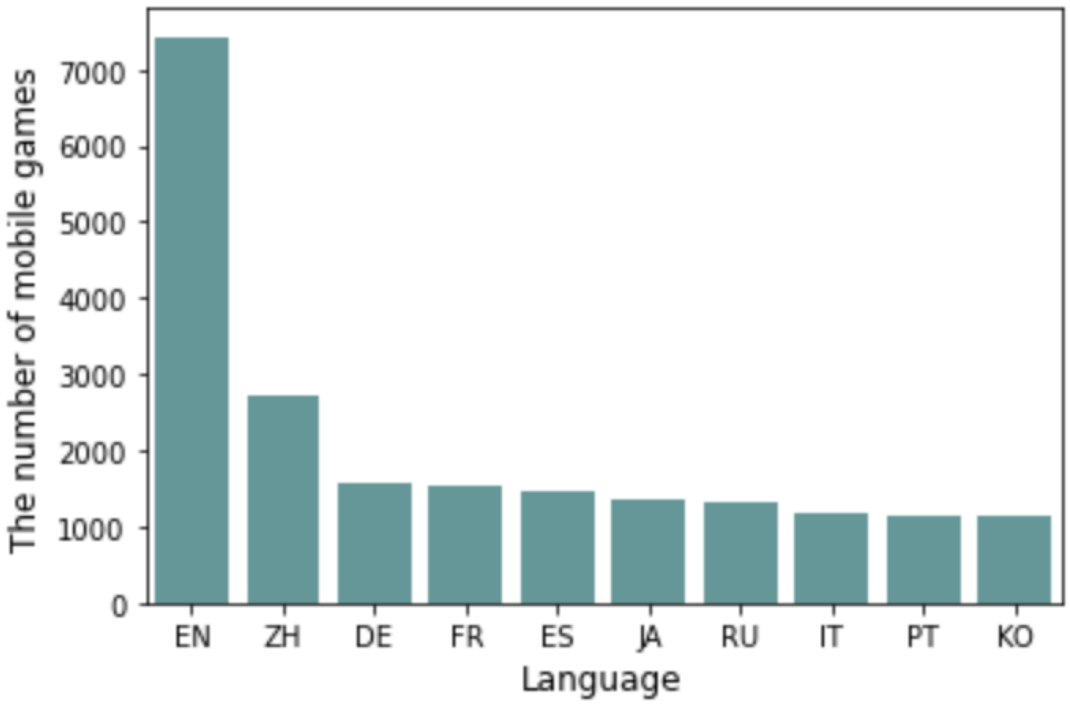


Abbildung 3.8: Häufigkeit der Sprachen in mobilen Games

Englisch liegt auf dem ersten Platz mit über 7.000 Spielen, während Chinesisch mit 3.000 Spielen den zweiten Platz einnimmt. Außerdem gibt noch viele andere Sprachen, die benutzt wurden, jedoch sehr selten vorkommen im Vergleich zu den genannten Sprachen.

Nun betrachten wir die **Multilingualität der Spiele**.

Sehr viele Spiele werden lediglich in einer Sprache veröffentlicht: Auf knapp 5000 Spiele trifft diese Aussage zu. Während die Anzahl der Spiele mit zwei oder mehr Sprachen sehr gering ist - lediglich 500 Spiele.

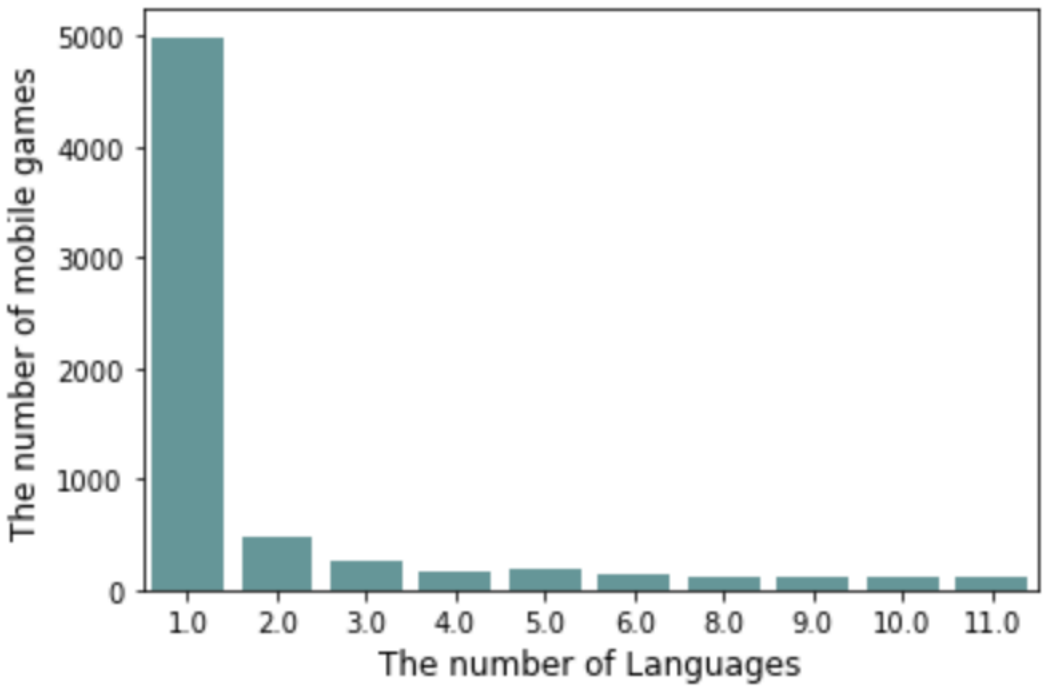


Abbildung 3.9: Multilingualität der Spiele

Die Anzahl von Neuveröffentlichungen der Handyspiele

Als nächstes wollte ich mir anschauen, wie sich die Menge der mobilen Games nach Monetarisierungsmodell der Apps im Laufe der Jahre verändert hat.

Im Allgemeinen nimmt die Anzahl der Spiele tendenziell zu. Die Menge der kostenfreien Games mit In-App Kaufoptionen wuchsen sehr schnell. Allein im Jahr 2016 wurden 600 Spiele mit diesem Monetarisierungsmodell veröffentlicht. Die komplett kostenfreie Spiele wuchsen auch, jedoch nicht so schnell. Die beiden anderen steigen sehr langsam an und verändern sich kaum.

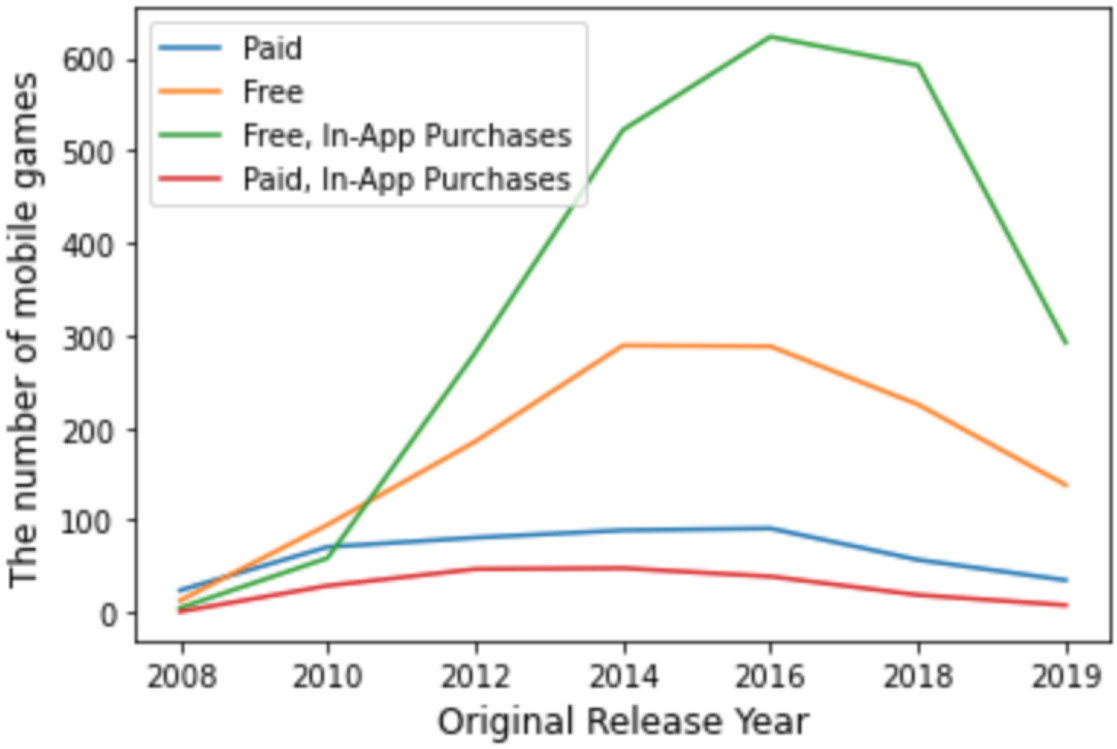


Abbildung 3.10: Anzahl der Neuveröffentlichungen der Spiele nach Monetarisierungsmodell

Das Wachstum der durchschnittlichen Nutzerbewertung der Handyspiele

Obwohl der mobile Spielemarkt allmählich gesättigt wirkt und die Anzahl der Neuerscheinungen nicht mehr so stark wächst, folgt die durchschnittliche Nutzerbewertung über die Jahre immer noch einem Aufwärtstrend. Auch die Dateigröße der Spiele nimmt weiterhin zu. Das kann bedeuten, dass die Qualität der Spiele vermutlich immer weiter verbessert wird. Zum Beispiel durch Fehleroptimierungen, schönere Benutzeroberflächen, Spielerweiterungen. Oder die Spiele laufen durch die leistungsfähigere Hardware der Benutzer besser und sorgen so für weniger Frustration bei den Spielern.

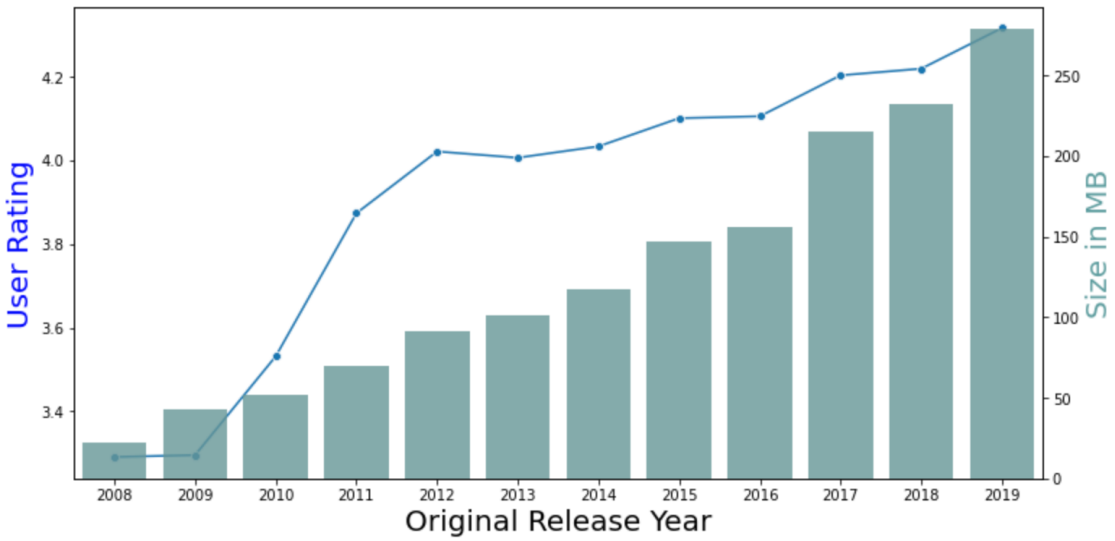


Abbildung 3.11: Wachstum der Nutzerbewertungen und der Dateigröße der Handyspiele im Laufe der Jahren

Korrelation zwischen Features

Die Heatmap plot zeigt uns die Korrelation zwischen den einzelnen Variablen.

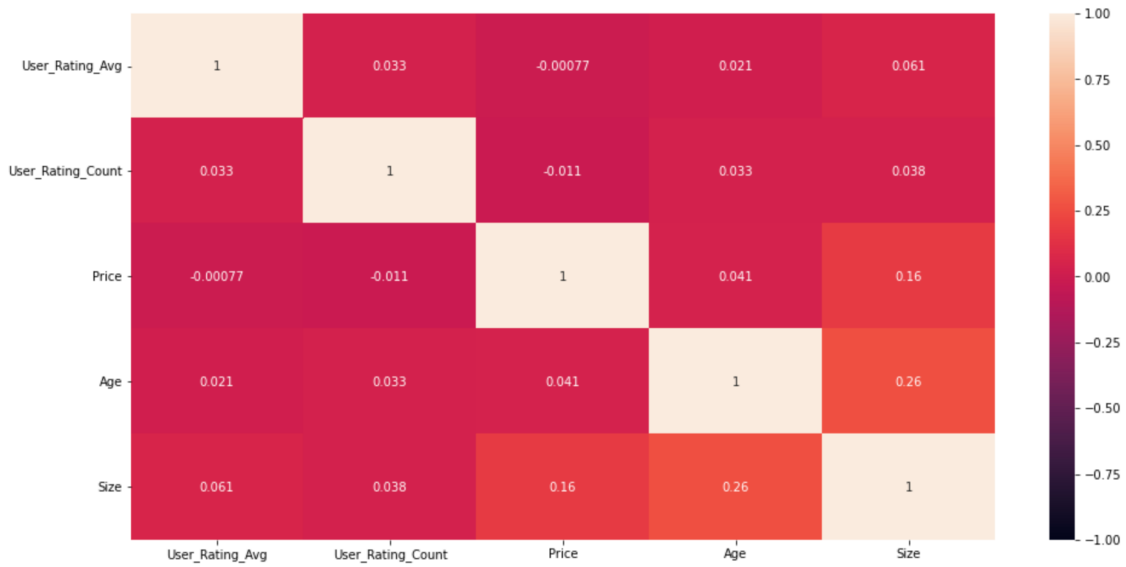


Abbildung 3.12: Korrelation der Features

In diesem Fall hat Average User Rating eine positive Korrelation mit User rating count, Age, und Size. Das bedeutet, mit steigendem Average User Rating nehmen die anderen Variablen auch zu.

Andersrum hat das Average User Rating eine negative Korrelation zur Variable Preis. Das sagt uns, dass bei einer Senkung des Preises die durchschnittliche Benutzerbewertung eines Spiels steigen könnte.

3.3 Modellierung - Vorhersage der durchschnittlichen Nutzerbewertung

Zuerst betrachten wir die **Binary Logistic Regression Model** mit der abhängigen Variable *User Rating Avg* (wenn die durchschnittliche Benutzerbewertung größer oder gleich 4.0 ist, ist sie gut und wird als 1 gewertet; andernfalls wird es als 0 bezeichnet).

```
# User Rating Average from 4.0 is good, refers to 1
# User Rating Average under 4.0 is bad, refers to 0
df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 5.0, "1", df1['User_Rating_Avg'])
df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 4.5, "1", df1['User_Rating_Avg'])
df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 4.0, "1", df1['User_Rating_Avg'])

df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 3.5, "0", df1['User_Rating_Avg'])
df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 3.0, "0", df1['User_Rating_Avg'])
df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 2.5, "0", df1['User_Rating_Avg'])
df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 2.0, "0", df1['User_Rating_Avg'])
df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 1.5, "0", df1['User_Rating_Avg'])
df1['User_Rating_Avg'] = np.where(df1['User_Rating_Avg'] == 1.0, "0", df1['User_Rating_Avg'])
```

Abbildung 3.13: Die Umstellung der abhängigen Variable in Python Codes.

Danke der *numpy* Bibliothek können wir sehr einfach Features auswählen. Die unerklärende Variablen sind *User Rating Count, Price, Age and Size* und die erklärende Variable ist *User Rating Avg*.
Mithilfe der Bibliothek *Sklearn* sind wir jetzt bereit für die Modellierungsphase. Zuerst müssen wir unsere Daten zufällig in zwei Teilmengen aufteilen:

- das Trainingsset zum Trainieren des Modells (30% des Datensatzes)
- das Testset zur Evaluation des Modells (70% des Datensatzes).

Dann verwenden wir den *StandardScaler* in Python um den Datensatz zu normalisieren. Außerdem müssen wir noch die Modelle und die wichtige Metriken die wir hier benutzen möchten erstellen.

Es gibt viele Metriken, mit denen man die Leistung eines Modells messen kann. Hier benutzen wir erstmal die Accuracy. Die Genauigkeit bzw. Accuracy beträgt 0.9666221628838452 bzw. 96,66% (das heißt es gab fast 97 richtige Vorhersagen aus 100 Beispielen). Das bedeutet, dass unser Vorhersage der durchschnittlichen Nutzerbewertung hervorragende Arbeit leistet. Einer der Gründe, warum die Genauigkeit, die wir erhalten, so hoch ist, liegt darin, dass dieser Datensatz eine ungleichmäßige Verteilung aufweist. Durch die obige Analyse können wir leicht erkennen, dass es nach rechts verzerrt ist.

Daneben wird die Konfusionmatrix auch benutzt um das Modell zu bewerten. Es gibt eine Tabelle, die die Genauigkeit der y_{test} und $y_{predict}$ zeigt:

y_{test}	$y_{predict}$
3.0	3.0
4.5	4.5
4.0	4.0
4.0	4.0
4.5	4.5

Tabelle 3.2: Die Genauigkeit der y_test und $y_predict$. Siehe Python Code.

Wir plottieren hier einen Heatmap der Konfusionmatrix.

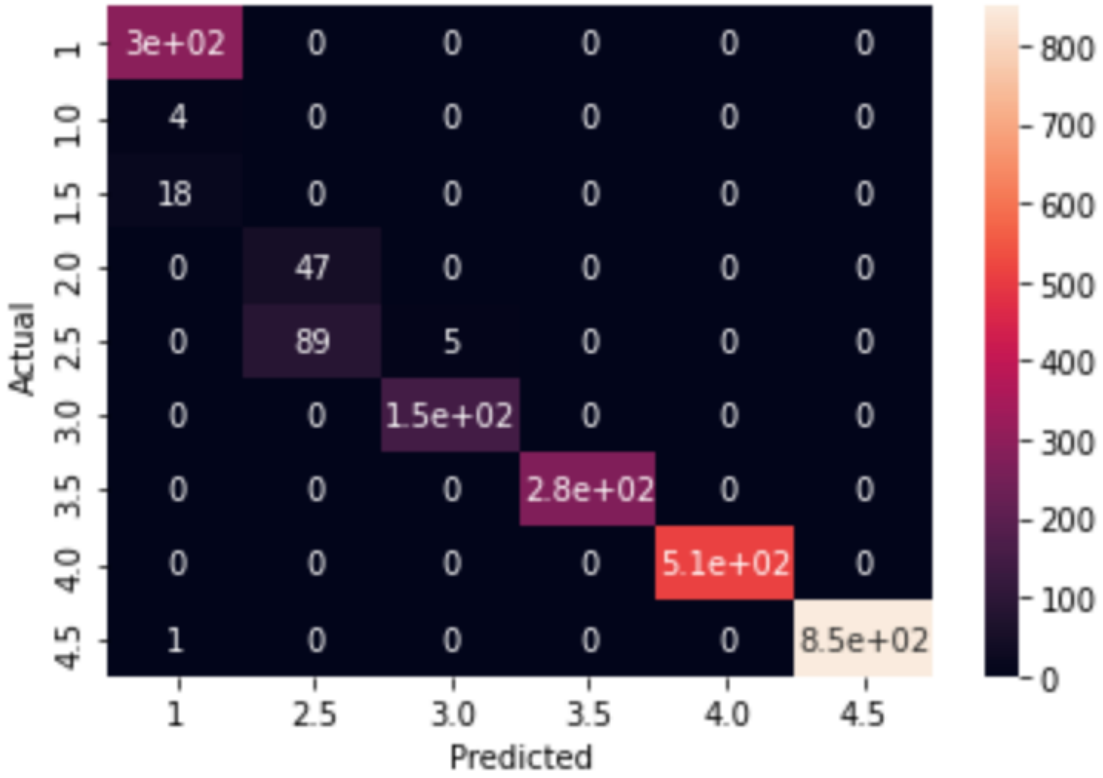


Abbildung 3.14: Konfusionmatrix

Laut der Grafik gibt es nur eine falsche Vorhersage als 1.0 bei einer realen Nutzerbewertung von 4.5. Das gleiche bei realen 2.0 Bewertungen wurden 47 Handyspiele falsche vorhergesagt.

Aus Python bekommen wir dann noch eine Classification-report:

Avg	Precision	Recall	F1-Score
1.0	0.93	1.00	0.96
1.0	0.00	0.00	0.00
1.5	0.00	0.00	0.00
2.0	0.00	0.00	0.00
2.5	0.65	0.95	0.77
3.0	0.97	1.00	0.98
3.5	1.00	1.00	1.00
4.0	1.00	1.00	1.00
4.5	1.00	1.00	1.00

Tabelle 3.3: Classification-Report. Siehe Python Code.

Somit ist ein Klassifikator mit Precision = Recall = 1.00 in diesem Maß besser als ein anderer Klassifikator mit Precision = 0.65 und Recall = 0,95.

Außerdem können wir hier noch das **Ordinal Logistic Regression Model** verwenden. Der Grund dafür ist, dass unser abhängige Variable *Average User Rating* eine ordinale Kategorie ist. Die Accuracy bei dieser Methode beträgt 1.0 bzw. 100%. So eine hohe Accuracy ist nicht immer ein gutes Zeichen. Bei einem Datensatz der z.B. wie in diesem Fall rechtsschief ist, kann die gute Accuracy täuschen.

4 Fazit und Ausblick

Zusammenfassend können wir folgendes Fazit ziehen: Die Anzahl der mobilen Games steigt weiterhin an, allerdings nicht mehr so viel wie in den Jahren 2014-2018: in denen die meisten Spiele herauskamen. Scheinbar gibt es einen Zusammenhang zwischen den Nutzerbewertungen und der Applikationsgröße. Aufwändigere Spiele, die mehr Speicherplatz verwenden kommen bei den Spielern generell besser an. Die wichtigsten Faktoren eines erfolgreichen Spiels sind zum Einen die Größe, die Altersfreigabe und zum anderen natürlich die Sprache in denen die Spiele veröffentlicht werden. Außerdem konnten wir eine leichte negative Korrelation zwischen dem Preis und der Benutzerbewertung feststellen.
Mein programmiertes Model konnte die Benutzerbewertungen mittels ordinal logistischer Regression am besten vorhersagen. Die Accuracy lag hier bei 0,998. Während sie bei der binär logistischen Regression bei ebenfalls mehr als guten 0,9675 lag.

Die logistische Regression ist eine gut geeignet um Klassifikationsprobleme zu lösen. Sie wird aus vielen verschiedenen Gründen häufig verwendet, ist jedoch nicht so einfach umzusetzen mit ihrer restriktiven Vorhersagekraft. Außerdem haben andere Modelle möglicherweise bessere Vorhersageergebnisse. In meinem Anwendungsbeispiel ist es das Logistic Regression Model nicht ideal gewählt, obwohl wir eine gute Accuracy bekommen haben. Außerdem können wir andere Methode zum Besipiel Randomforest oder Support Vector Maschine verwenden. Ein weiterer Nachteil des logistischen Regressionsmodells besteht darin, dass die Interpretation schwieriger ist, da die Interpretation der Gewichte multiplikativ und nicht additiv ist. Im Gegenteil zum logistischen Regressionsmodell hat das aber auch seine Vorteile. Es gibt uns nicht nur ein Klassifizierungsmodell sondern auch Wahrscheinlichkeiten. Das ist ein großer Vorteil gegenüber Modellen, die nur die endgültige Einordnung liefern können. Zu wissen, dass eine Instanz eine Wahrscheinlichkeit von 90% für eine Klasse hat, im Vergleich zu 40%, macht einen großen Unterschied. Die logistische Regression kann auch von der binären Klassifikation auf die Mehrklassenklassifikation erweitert werden. Dann

heißt es multinomiale Regression. Sowohl maschinelles Lernen eine sehr interessantes Themengebiet ist, ist auch noch ein sehr neues und nicht triviales Gebiet, was sich in der Zukunft noch sehr stark weiterentwickeln wird.

Literaturverzeichnis

Skript in der Vorlesung

[1] Statistical Learning - Vorlesung von Prof. Dr. Christina Erlwein-Sayer in SoSe2021

[2] Statistik 3 - Vorlesung von Prof. Dr. Dietmar Hillebrand in WS21/22

Quelle:

https://www.kaggle.com/tristan581/datasets
https://www.game.de/deutscher-markt-fuer-mobile-games-waechst-um-23-prozent/
https://rikunert.com/ordinal_rating
https://studyflix.de/statistik/logistische-regression-2150
https://blog.codecentric.de/2019/07/machine-learning-modelle-bewerten-die-crux-mit-
https://machinelearningcoban.com/2017/08/31/evaluation/?fbclid=IwAR0ekfy_
2nNAh0LsTzPBklCTqG0rcTzAfuR9ibngDMneetLO-TjAxCZFSlk#-precision-va-recall

Bücher:

[1] Richter, S.(2019): Statistisches und maschinelles Lernen, Springer, Heidelberg

[2] Christoph Molnar, S.(2019): Interpretable Machine Learning - A Guide for Making Black Box Models Explainable - https://christophm.github.io/interpretable-ml-book/

Abbildungsverzeichnis

2.1 Linear Regression vs. Logistic Regression 5
2.2 Precision und Recall 11

3.1 Top 5 Games der Datensatzes 13
3.2 Nutzerbewertung vs. Anzahl der Nutzerbewertungen 14
3.3 Anzahl der mobilen Games vs. Monetarisierung 15
3.4 Durchschnittliche Nutzerbewertung vs. Monetarisierung 16
3.5 Anzahl der Games nach Monetarisierungsmodell 16
3.6 Benutzerbewertung vs. Alter 17
3.7 Beziehung zwischen Size, Preis und Nutzerbewertung 18
3.8 Häufigkeit der Sprachen in mobilen Games 20
3.9 Multilingualität der Spiele 21
3.10 Anzahl der Neuveröffentlichungen der Spiele nach Monetarisierungsmodell 22
3.11 Wachstum der Nutzerbewertungen und der Dateigröße der Handyspiele im Laufe der Jahren 23
3.12 Korrelation der Features 24
3.13 Die Umstellung der abhängigen Variable in Python Codes. 25
3.14 Konfusionmatrix 26

Tabellenverzeichnis

3.1 Die Tabelle zeigt die Anzahl der mobilen Games nach Primary Genre. Siehe das in Python-Skript. 19
3.2 Die Genauigkeit der y_test und $y_predict$. Siehe Python Code. 26
3.3 Classification-Report. Siehe Python Code. 27

Selbstständigkeitserklärung

I hereby confirm that I have authored this Seminar paper independently and without use of others than the indicated sources. All passages (and codes) which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, 27. Februar 2022

Ngoc Mai Dinh