

## APPENDIX

```

1. import pandas as pd
2. from pandasgui import show
3. import numpy as np
4. import string
5. import re
6.
7. import nltk
8. from nltk.corpus import wordnet
9. from nltk.stem import WordNetLemmatizer
10.
11. import matplotlib.pyplot as plt
12. import random as rd
13.
14. dataset = pd.read_csv("NewData/review_dataset/steam_reviews.csv")
15.
16.
17. # add new column 'Review Length' and 'Sum Recommendation'
18. dataset['review_length'] = dataset.apply(lambda row:
len(str(row['review']))), axis=1)
19. dataset['sum_recommendation'] = dataset['recommendation'] ==
'Recommended'
20. dataset['sum_recommendation'] =
dataset['sum_recommendation'].astype(int)
21. # Removing Null object
22. print("Before removing Nan : ", len(dataset))
23. clean_data = dataset.dropna()
24. clean_data = clean_data.reset_index(drop = True)
25. print("After removing Nan : ", len(clean_data))
26. print("Total Null Data : ", len(dataset)-len(clean_data))
27. # -----punctuation removal-----
-----
28.
29. def remove_punctuation(text):
30.     for punctuation in string.punctuation:
31.         text = text.replace(punctuation, '')
32.     return text
33.
34. # -----Stopwords Removal-----
-----
35.
36. stop = ["a", "about", "above", "after", "again", "against", "ain",
"all", "am", "an", "and", "any", "are",
37.         "aren", "aren't", "as", "at", "be", "because", "been", "before",
"being", "below", "between", "both",
38.         "but", "by", "can", "couldn", "couldn't", "d", "did", "didn",
"didn't", "do", "does", "doesn", "doesn't",
39.         "doing", "don", "don't", "down", "during", "each", "few", "for",
"from", "further", "had", "hadn", "hadn't",
40.         "has", "hasn", "hasn't", "have", "haven", "haven't", "having",
"he", "her", "here", "hers", "herself", "him",
41.         "himself", "his", "how", "i", "if", "in", "into", "is", "isn",
"isn't", "it", "it's", "its", "itself", "just",

```

42. "ll", "m", "ma", "me", "mightn", "mightn't", "more", "most",  
"mustn", "mustn't", "my", "myself", "needn", "needn't",
43. "no", "nor", "not", "now", "o", "of", "off", "on", "once",  
"only", "or", "other", "our", "ours", "ourselves", "out",
44. "over", "own", "re", "s", "same", "shan", "shan't", "she",  
"she's", "should", "should've", "shouldn", "shouldn't",
45. "so", "some", "such", "t", "than", "that", "that'll", "the",  
"their", "theirs", "them", "themselves", "then", "there",
46. "these", "they", "this", "those", "through", "to", "too",  
"under", "until", "up", "ve", "very", "was", "wasn", "wasn't",
47. "we", "were", "weren", "weren't", "what", "when", "where",  
"which", "while", "who", "whom", "why", "will", "with", "won",
48. "won't", "wouldn", "wouldn't", "y", "you", "you'd", "you'll",  
"you're", "youre", "you've", "your", "yours", "yourself", "yourselves",
49. "could", "he'd", "he'll", "he's", "here's", "how's", "i'd",  
"i'll", "i'm", "im", "ive", "i've", "let's", "lets", "ought", "she'd",  
"she'll",
50. "that's", "there's", "they'd", "they'll", "they're", "they've",  
"we'd", "we'll", "we're", "we've", "weve", "what's", "when's",
51. "where's", "who's", "why's", "would", "able", "abst",  
"accordance", "according", "accordingly", "across", "act",
52. "actually", "added", "adj", "affected", "affecting", "affects",  
"afterwards", "ah", "almost", "alone", "along",
53. "already", "also", "although", "always", "among", "amongst",  
"announce", "another", "anybody", "anyhow", "anymore",
54. "anyone", "anything", "anyway", "anyways", "anywhere",  
"apparently", "approximately", "arent", "arise", "around",
55. "aside", "ask", "asking", "auth", "available", "away",  
"awfully", "b", "back", "became", "become", "becomes",
56. "becoming", "beforehand", "begin", "beginning", "beginnings",  
"begins", "behind", "believe", "beside", "besides",
57. "beyond", "biol", "brief", "briefly", "c", "ca", "came",  
"cannot", "can't", "cause", "causes", "certain", "certainly",
58. "co", "com", "come", "comes", "contain", "containing",  
"contains", "couldnt", "date", "different", "done", "downwards",
59. "due", "e", "ed", "edu", "effect", "eg", "eight", "eighty",  
"either", "else", "elsewhere", "end", "ending", "enough", "especially",
60. "et", "etc", "even", "ever", "every", "everybody", "everyone",  
"everything", "everywhere", "ex", "except", "f", "far", "ff",
61. "fifth", "first", "five", "fix", "followed", "following",  
"follows", "former", "formerly", "forth", "found", "four",
62. "furthermore", "g", "gave", "get", "gets", "getting", "give",  
"given", "gives", "giving", "go", "goes", "gone", "got",
63. "gotten", "h", "happens", "hardly", "hed", "hence",  
"hereafter", "hereby", "herein", "heres", "hereupon", "hes", "hi", "hid",  
"hither",
64. "home", "howbeit", "however", "hundred", "id", "ie", "im",  
"immediate", "immediately", "importance", "important", "inc",
65. "indeed", "index", "information", "instead", "invention",  
"inward", "itd", "it'll", "j", "k", "keep", "keeps", "kept",
66. "kg", "km", "know", "known", "knows", "l", "largely", "last",  
"lately", "later", "latter", "latterly",
67. "least", "less", "lest", "let", "lets", "like", "liked",  
"likely", "line", "little", "ll", "look",
68. "looking", "looks", "ltd", "made", "mainly", "make", "makes",  
"many", "may", "maybe", "mean", "means",

69. "meantime", "meanwhile", "merely", "mg", "might", "million",  
"miss", "ml", "moreover", "mostly", "mr",
70. "mrs", "much", "mug", "must", "n", "na", "name", "namely",  
"nay", "nd", "near", "nearly", "necessarily", "necessary",
71. "need", "needs", "neither", "never", "nevertheless", "new",  
"next", "nine", "ninety", "nobody", "non", "none", "nonetheless",
72. "noone", "normally", "nos", "noted", "nothing", "nowhere",  
"obtain", "obtained", "obviously", "often", "oh", "ok", "okay",
73. "old", "omitted", "one", "ones", "onto", "ord", "others",  
"otherwise", "outside", "overall", "owing", "p", "page", "pages",
74. "part", "particular", "particularly", "past", "per", "perhaps",  
"placed", "please", "plus", "poorly", "possible", "possibly",  
"potentially",
75. "pp", "predominantly", "present", "previously", "primarily",  
"probably", "promptly", "proud", "provides", "put", "q", "que",  
"quickly",
76. "quite", "qv", "r", "ran", "rather", "rd", "readily", "really",  
"recent", "recently", "ref", "refs", "regarding", "regardless",
77. "regards", "related", "relatively", "research", "respectively",  
"resulted", "resulting", "results", "right", "run",
78. "said", "saw", "say", "saying", "says", "sec", "section", "see",  
"seeing", "seem", "seemed", "seeming", "seems", "seen",
79. "self", "selves", "sent", "seven", "several", "shall", "shed",  
"shes", "show", "showed", "shown", "shows", "shows",
80. "significant", "significantly", "similar", "similarly",  
"since", "six", "slightly", "somebody", "somehow", "someone",
81. "somethan", "something", "sometime", "sometimes", "somewhat",  
"somewhere", "soon", "sorry", "specifically", "specified",
82. "specify", "specifying", "still", "stop", "strongly", "sub",  
"substantially", "successfully", "sufficiently", "suggest",
83. "sup", "sure", "take", "taken", "taking", "tell", "tends", "th",  
"thank", "thanks", "thanx", "thats", "that've", "thence", "thereafter",  
"thereby", "thered", "therefore", "therein", "there'll", "thereof",  
"therere", "theres", "thereto", "thereupon", "there've", "theyd",  
"theyre", "think", "thou", "though", "thoughh", "thousand", "throug",  
"throughout", "thru", "thus", "til", "tip", "together", "took", "toward",  
"towards", "tried", "tries", "truly", "try", "trying", "ts", "twice",  
"two", "u", "un", "unfortunately", "unless", "unlike", "unlikely",  
"unto", "upon", "ups", "us", "use", "used", "useful", "usefully",  
"usefulness", "uses", "using", "usually", "v", "value", "various", "'ve",  
"via", "viz", "vol", "vols",
84. "vs", "w", "want", "wants", "wasnt", "way", "wed", "welcome",  
"went", "werent", "whatever", "what'll", "whats", "whence", "whenever",  
"whereafter", "whereas", "whereby", "wherein", "wheres", "whereupon",  
"wherever", "whether", "whim", "whither", "whod", "whoever", "whole",  
"who'll", "whomever", "whos", "whose", "widely", "willing", "wish",  
"within", "without", "wont", "words", "world", "wouldnt", "www", "x",  
"yes", "yet", "youd", "youre", "z", "zero", "a's", "ain't", "allow",  
"allows", "apart", "appear", "appreciate", "appropriate", "associated",  
"best", "better", "c'mon", "c's", "cant", "changes", "clearly",  
"concerning", "consequently", "consider", "considering", "corresponding",  
"course", "currently", "definitely", "described", "despite", "entirely",  
"exactly", "example", "going", "greetings", "hello", "help", "hopefully",  
"ignored", "inasmuch", "indicate", "indicated", "indicates", "inner",  
"insofar", "it'd", "keep", "keeps", "novel", "presumably", "reasonably",  
"second", "secondly", "sensible", "serious", "seriously", "sure", "t's",

"third", "thorough", "thoroughly", "three", "well", "wonder", "a",  
 "about", "above", "above", "across", "after", "afterwards", "again",  
 "against", "all", "almost", "alone", "along", "already", "also",  
 "although", "always", "am", "among", "amongst", "amongst", "amount",  
 "an", "and", "another", "any", "anyhow", "anyone", "anything", "anyway",  
 "anywhere", "are", "around", "as", "at",  
 85. "back", "be", "became", "because", "become", "becomes",  
 "becoming", "been", "before", "beforehand", "behind", "being", "below",  
 "beside", "besides", "between", "beyond", "bill", "both", "bottom",  
 "but", "by", "call", "can", "cannot", "cant", "co", "con", "could",  
 "couldnt", "cry", "de", "describe", "detail", "do", "done", "down", "due",  
 "during", "each", "eg", "eight", "either", "eleven", "else", "elsewhere",  
 "empty", "enough", "etc", "even", "ever", "every", "everyone",  
 "everything", "everywhere", "except", "few", "fifteen", "fifty", "fill",  
 "find", "fire", "first", "five", "for", "former", "formerly", "forty",  
 "found", "four", "from", "front", "full", "further", "get", "give", "go",  
 "had", "has", "hasnt", "have", "he", "hence", "her", "here", "hereafter",  
 "hereby", "herein", "hereupon", "hers", "herself", "him", "himself",  
 "his", "how", "however", "hundred", "ie", "if", "in", "inc", "indeed",  
 "interest", "into", "is", "it", "its", "itself", "keep", "last", "latter",  
 "latterly", "least", "less", "ltd", "made", "many", "may", "me",  
 "meanwhile", "might", "mill", "mine", "more", "moreover", "most",  
 "mostly", "move", "much", "must", "my", "myself", "name", "namely",  
 "neither", "never", "nevertheless", "next", "nine", "no", "nobody",  
 "none", "noone", "nor", "not", "nothing", "now", "nowhere", "of", "off",  
 "often", "on", "once", "one", "only", "onto", "or", "other", "others",  
 "otherwise", "our", "ours", "ourselves", "out", "over", "own", "part",  
 "per", "perhaps", "please", "put", "rather", "re", "same", "see", "seem",  
 "seemed", "seeming", "seems", "serious", "several", "she", "should",  
 "show", "side", "since", "sincere", "six", "sixty", "so", "some",  
 "somehow", "someone", "something", "sometime", "sometimes", "somewhere",  
 "still", "such", "system", "take", "ten", "than", "that", "the", "their",  
 "them", "themselves", "then", "thence", "there", "thereafter", "thereby",  
 "therefore", "therein", "thereupon", "these", "they", "thickv", "thin",  
 "third", "this", "those", "though", "three", "through", "throughout",  
 "thru", "thus", "to", "together", "too", "top", "toward", "towards",  
 "twelve", "twenty", "two", "un", "under", "until", "up", "upon", "us",  
 "very", "via", "was", "we", "well", "were", "what", "whatever", "when",  
 "whence", "whenever", "where", "whereafter", "whereas", "whereby",  
 "wherein", "whereupon", "wherever", "whether", "which", "while",  
 "whither", "who", "whoever", "whole", "whom", "whose", "why", "will",  
 "with", "within", "without", "would", "yet", "you", "your", "yours",  
 "yourself", "yourselves", "the", "a", "b", "c", "d", "e", "f", "g", "h",  
 "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v",  
 "w", "x", "y", "z", "A", "B", "C", "D", "E", "F", "G", "H", "I", "J",  
 "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X",  
 "Y", "Z", "co", "op", "research-articl", "pagecount", "cit", "ibid",  
 "les", "le", "au", "que", "est", "pas", "vol", "el", "los", "pp", "u201d",  
 "well-b", "http", "volumtype", "par", "0o", "0s", "3a", "3b", "3d", "6b",  
 "6o", "a1", "a2", "a3", "a4", "ab", "ac", "ad", "ae", "af", "ag", "aj",  
 "al", "an", "ao", "ap", "ar", "av", "aw", "ax", "ay", "az", "b1", "b2",  
 "b3", "ba", "bc", "bd", "be", "bi", "bj", "bk", "bl", "bn", "bp", "br",  
 "bs", "bt", "bu", "bx", "c1", "c2", "c3", "cc", "cd", "ce", "cf", "cg",  
 "ch", "ci", "cj", "cl", "cm", "cn", "cp", "cq", "cr", "cs", "ct", "cu",  
 "cv", "cx", "cy", "cz", "d2", "da", "dc", "dd", "de", "df", "di", "dj",  
 "dk", "dl", "do", "dp", "dr", "ds", "dt", "du", "dx", "dy", "e2", "e3",

```

"ea", "ec", "ed", "ee", "ef", "ei", "ej", "el", "em", "en", "eo", "ep",
"eq", "er", "es", "et", "eu", "ev", "ex", "ey", "f2", "fa", "fc", "ff",
"fi", "fj", "fl", "fn", "fo", "fr", "fs", "ft", "fu", "fy", "ga", "ge",
"gi", "gj", "gl", "go", "gr", "gs", "gy", "h2", "h3", "hh", "hi", "hj",
"ho", "hr", "hs", "hu", "hy", "i", "i2", "i3", "i4", "i6", "i7", "i8",
"ia", "ib", "ic", "ie", "ig", "ih", "ii", "ij", "il", "in", "io", "ip",
"iq", "ir", "iv", "ix", "iy", "iz", "jj", "jr", "js", "jt", "ju", "ke",
"kg", "kj", "km", "ko", "l2", "la", "lb", "lc", "lf", "lj", "ln", "lo",
"lr", "ls", "lt", "m2", "ml", "mn", "mo", "ms", "mt", "mu", "n2", "nc",
"nd", "ne", "ng", "ni", "nj", "nl", "nn", "nr", "ns", "nt", "ny", "oa",
"ob", "oc", "od", "of", "og", "oi", "oj", "ol", "om", "on", "oo", "oq",
"or", "os", "ot", "ou", "ow", "ox", "oz", "p1", "p2", "p3", "pc", "pd",
"pe", "pf", "ph", "pi", "pj", "pk", "pl", "pn", "po", "pq", "pr",
"ps", "pt", "pu", "py", "qj", "qu", "r2", "ra", "rc", "rd", "rf", "rh",
"ri", "rj", "rl", "rm", "rn", "ro", "rq", "rr", "rs", "rt", "ru", "rv",
"ry", "s2", "sa", "sc", "sd", "se", "sf", "si", "sj", "sl", "sm", "sn",
"sp", "sq", "sr", "ss", "st", "sy", "sz", "t1", "t2", "t3", "tb", "tc",
86.     "td", "te", "tf", "th", "ti", "tj", "tl", "tm", "tn", "tp", "tq",
"tr", "ts", "tt", "tv", "tx", "ue", "ui", "uj", "uk", "um", "un", "uo",
"ur", "ut", "va", "wa", "vd", "wi", "vj", "vo", "wo", "vq", "vt", "vu",
"xl", "x2", "x3", "xf", "xi", "xj", "xk", "xl", "xn", "xo", "xs",
87.     "xt", "xv", "xx", "y2", "yj", "yl", "yr", "ys", "yt", "zi",
"zz"]
88.
89. # print(len(stop))
90.
91. def stopwords(text):
92.     return " ".join([word for word in str(text).split() if word not in
stop])
93.
94. # -----Emoticon Removal-----
-----
95.
96. from emot.emo_unicode import UNICODE_EMO, EMOTICONS
97.
98. def remove_emoticons(text):
99.     emoticon_pattern = re.compile(u'(' + u'|'.join(k for k in EMOTICONS)
+ u')')
100.     return emoticon_pattern.sub(r'', text)
101.
102. # -----URL Removal-----
-----
103.
104. def remove_url(text):
105.     utl_prattern = re.compile(r'https?://\S+|www\.\S+')
106.     return url_pattern.sub(r'', text)
107.
108. # -----Word Lemmatizer-----
-----
109.
110. lemmatizer = WordNetLemmatizer()
111. wordnet_map = {"N":wordnet.NOUN, "V":wordnet.VERB, "J":wordnet.ADJ,
"R":wordnet.ADV} #noun, verb, adjective, and adverb
112.
113. def lemmatize_word(text):
114.     pos_tagged_text = nltk.pos_tag(text.split())

```

```

115.     return " ".join([lemmatizer.lemmatize(word, wordnet_map.get(pos[0],
    wordnet.NOUN)) for word, pos in pos_tagged_text])
116.
117. # -----Word Tokenizer-----
    -----
118.
119. def tokenize(text):
120.     text = re.split('\W+', text)
121.     return text
122. # Data Initiation
123.
124. # change inset to set how much data inserted
125. inset = 500
126.
127. train = clean_data['review'].head(inset)
128. helpful = clean_data['helpful'].head(inset)
129. hour_play = clean_data['hour_played'].head(inset)
130. length = clean_data['review_length'].head(inset)
131. sum_rec = clean_data['sum_recommendation'].head(inset)
132. # Runnning Preprocessing
133. train = train.apply(stopwords)
134. train = train.apply(remove_punctuation)
135. train = train.str.lower()
136. train = train.apply(remove_emoticons)
137. train = train.apply(lemmatize_word)
138. train = train.apply(stopwords)
139. train_tokenize = train.apply(tokenize)
140.
141. print("Preprocessing Done")
142. # Deleting data on other column that have whitespace review
143.
144. helpful_list = helpful.tolist()
145. hour_list = hour_play.tolist()
146. sum_list = sum_rec.tolist()
147. length_list = length.tolist()
148.
149. # index of each data need to be removed
150. for i in range(len(train)):
151.     if train[i] == '':
152.         try:
153.             print(i)
154.         except KeyError:
155.             print("This index not existed")
156.
157. for x in range(len(train)):
158.     if train[x] == '':
159.         try:
160.             del hour_list[x]
161.         except IndexError:
162.             print("This index ",x," not existed in hour list")
163.
164. for x in range(len(train)):
165.     if train[x] == '':
166.         try:
167.             del helpful_list[x]
168.         except IndexError:

```

```

169.         print("This index ",x," not existed in helpful ")
170.
171. for x in range(len(train)):
172.     if train[x] == '':
173.         try:
174.             del sum_list[x]
175.         except IndexError:
176.             print("This index ",x," not existed in sum rec")
177.
178. for x in range(len(train)):
179.     if train[x] == '':
180.         try:
181.             del length_list[x]
182.         except IndexError:
183.             print("This index ",x," not existed in text length")
184.
185. print("Length list ",len(length_list))
186. print("Hour List ",len(hour_list))
187. print("Helpful List ",len(helpful_list))
188. print("Sum Rec List ",len(sum_list))
189. train = [x for x in train if x != '']
190. listed = list(filter(None, train_tokenize))
191. removed_comp = [x for x in listed if x != ['']]
192. removed_comp = [x for x in removed_comp if x != []]
193. word_frequency = {}
194. for i in range(len(removed_comp)):
195.     tokens = removed_comp[i]
196.     for word in range(len(tokens)):
197.         snap = tokens[word]
198.         try:
199.             word_frequency[snap].add(i)
200.         except:
201.             word_frequency[snap] = {i}
202.
203. DF = {}
204. for i in word_frequency:
205.     DF[i] = len(word_frequency[i])
206.
207. total_vocab = [x for x in DF]
208. print(len(total_vocab))
209. idf = {}
210. N = len(train) #total data inserted
211.
212. for token in total_vocab:
213.     data_freq = DF[token]
214.     idf[token] = np.log10(len(train)/(data_freq)) #counting idf = total
        review/DF[i]
215.
216. print('total words is : ',len(idf))
217. import time
218. t0 = time.perf_counter()
219. tf = {}
220. for token in total_vocab:
221.     vector_tf = []
222.     appears = word_frequency[token]
223.     for document in range(len(removed_comp)):

```

```

224.         doc_freq = 0
225.         for word in nltk.word_tokenize(train[document]):
226.             if token == word:
227.                 doc_freq += 1
228.                 word_tf = doc_freq/len(removed_comp)
229.                 vector_tf.append(word_tf)
230.             tf[token] = vector_tf
231.
232. from pandasgui import show
233. tf
234. term_freq_df = pd.DataFrame(data = tf)
235. t1 = time.perf_counter()
236. print(f'TF counter done in {t1-t0:0.4f} seconds')
237. t0 = time.perf_counter()
238. tfidf = []
239. for token in tf.keys():
240.     sentence_tfidf = []
241.     for tf_sentence in tf[token]:
242.         tfidf_score = tf_sentence * idf[token]
243.         sentence_tfidf.append(tfidf_score)
244.     tfidf.append(sentence_tfidf)
245.
246. weight = []
247. for i in range(0, len(tfidf[0])):
248.     tmp = 0
249.     for j in range(0, len(tfidf)):
250.         tmp = tmp + tfidf[j][i]
251.     weight.append(tmp)
252.
253. # weight
254. weight_df = pd.DataFrame(data = weight, columns=['Weight'])
255. t1 = time.perf_counter()
256. print(f'TF-IDF counter done in {t1-t0:0.2f} seconds')
257. df_length = pd.DataFrame(data = length_list, columns = ['Text Length'])
258.     # df for text length data
259.
260. df_hour = pd.DataFrame(data = hour_list, columns = ['Hour Played']) #
261.     df for hour of play data
262.
263. df_helpful = pd.DataFrame(data = helpful_list, columns = ['Helpful']) #
264.     df for helpful data
265.
266. df_reco = pd.DataFrame(data = sum_list, columns = ['Recommended']) # df
267.     for sum recommendation data
268.
269. # -----joined dataframe-----
270.
271. df_row_length = weight_df.join(df_length) # df for weight and text
272.     length
273. df_row_length = df_row_length.dropna()
274.

```

```

275. df_row_hour = weight_df.join(df_hour) # df for weight and hour of play
276. df_row_hour = df_row_hour.dropna()
277.
278.
279. df_row_helpful =weight_df.join(df_helpful) # df for weight and helpful
280. df_row_helpful = df_row_helpful.dropna()
281.
282.
283. df_corr = df_helpful.join(df_hour) # df for helpful and hour of play
284. df_corr = df_corr.dropna()
285.
286. df_rec = weight_df.join(df_reco) # df for weight and sum recommendation
287. df_rec = df_rec.dropna()
288.
289. df_rec_help = df_helpful.join(df_reco) # df for helpful and sum
      recommendation
290. df_rec_help = df_rec_help.dropna()
291.
292. df_rec_hour = df_hour.join(df_reco) # df for hour of play and sum
      recommendation
293. df_rec_hour = df_rec_hour.dropna()
294. # only activate one variable to be able to use it
295. # use '#'
296.
297. # ----- helpful and weight -----
298. # x = np.round(df_row_helpful.iloc[:,[0,1]].values, 2)
299.
300. # ----- hour of play and weight -----
301. # x = np.round(df_row_hour.iloc[:,[0,1]].values, 2)
302.
303. # ----- weight and sum recommendation -----
304. # x = np.round(df_rec.iloc[:,[0,1]].values, 2)
305. x = np.round(df_rec.reset_index().values,2)
306.
307. # ----- helpful and sum recommendation -----
308. # x = df_rec_help.iloc[:,[0,1]].values
309.
310. # ----- hour of play and sum recommendation -----
311. # x = df_rec_hour.iloc[:,[0,1]].values
312.
313. # ----- text length and weight -----
314. # x = np.round(df_row_length.iloc[:,[0,1]].values, 2)
315. m = x.shape[0]
316. n = x.shape[1]
317. n
318.
319. l = x.T
320. l = l[1:].T
321.
322. plt.scatter(x[:,1],x[:,2],c='black',label='unclustered data')
323. plt.xlabel('Weight')
324. plt.ylabel('Text Length')
325. plt.legend()
326. plt.title('Plot of data points')
327. plt.show()
328. K = 2

```

```

329. Centroids = np.array([]).reshape(n,0)
330. for i in range(K):
331.     rand = rd.randint(0,m-1)
332.     print(rand)
333.     Centroids = np.c_[Centroids, x[rand]]
334.
335. print(Centroids)
336.
337. Output = {}
338. count = 0
339.
340. while True:
341.     EuclidianDistance = np.array([]).reshape(m,0)
342.     for k in range(K):
343.         tempDist = np.sum((1-Centroids[1:,k])**2,axis=1)
344.         EuclidianDistance = np.c_[EuclidianDistance, tempDist]
345.     C = np.argmin(EuclidianDistance, axis=1)+1
346.     Y = {}
347.     for k in range(K):
348.         Y[k+1] = np.array([]).reshape(3,0)
349.
350.     for i in range(m):
351.         Y[C[i]] = np.c_[Y[C[i]], x[i]]
352.
353.     for k in range(K):
354.         Y[k+1] = Y[k+1].T
355.
356.     for k in range(K):
357.         Centroids[:, k] = np.mean(Y[k+1],axis=0)
358.
359.
360.     if Output == {}:
361.         Output = Y
362.         count += 1
363.         print('First Iteration')
364.     else:
365.         if K == 2:
366.             if np.array_equal(Output[1], Y[1]) == True:
367.                 if np.array_equal(Output[2], Y[2]) == True:
368.                     if (Output[1] == Y[1]).all() == True:
369.                         if (Output[2] == Y[2]).all() == True:
370.                             print("Operation Done with : ", count, "
Iteration")
371.                                 break
372.                             else:
373.                                 Output = Y
374.                                 count += 1
375.                             else:
376.                                 Output = Y
377.                                 count += 1
378.                             else:
379.                                 Output[2] = Y[2]
380.                                 count += 1
381.                             else:
382.                                 Output[1] = Y[1]
383.                                 count += 1

```

```

384.         elif K == 3:
385.             if np.array_equal(Output[1], Y[1]) == True:
386.                 if np.array_equal(Output[2], Y[2]) == True:
387.                     if np.array_equal(Output[3], Y[3]) == True:
388.                         if (Output[1] == Y[1]).all() == True:
389.                             if (Output[2] == Y[2]).all() == True:
390.                                 if (Output[3] == Y[3]).all() == True:
391.                                     print("Operation Done with : ",
count, " Iteration")
392.                                         break
393.                                         else:
394.                                             Output = Y
395.                                             count += 1
396.                                         else:
397.                                             Output = Y
398.                                             count += 1
399.                                         else:
400.                                             Output = Y
401.                                             count += 1
402.                                         else:
403.                                             Output[3] = Y[3]
404.                                             count += 1
405.                                         else:
406.                                             Output[2] = Y[2]
407.                                             count += 1
408.                                         else:
409.                                             Output[1] = Y[1]
410.                                             count += 1
411.         elif K == 4:
412.             if np.array_equal(Output[1], Y[1]) == True:
413.                 if np.array_equal(Output[2], Y[2]) == True:
414.                     if np.array_equal(Output[3], Y[3]) == True:
415.                         if np.array_equal(Output[4], Y[4]) == True:
416.                             if (Output[1] == Y[1]).all() == True:
417.                                 if (Output[2] == Y[2]).all() == True:
418.                                     if (Output[3] == Y[3]).all() ==
True:
419.                                         if (Output[4] == Y[4]).all() ==
True:
420.                                             print("Operation Done with
: ", count, " Iteration")
421.                                             break
422.                                             else:
423.                                                 Output = Y
424.                                                 count += 1
425.                                             else:
426.                                                 Output = Y
427.                                                 count += 1
428.                                             else:
429.                                                 Output = Y
430.                                                 count += 1
431.                                             else:
432.                                                 Output = Y
433.                                                 count += 1
434.                                         else:
435.                                             Output[4] = Y[4]

```

```

436.             count += 1
437.         else:
438.             Output[3] = Y[3]
439.             count += 1
440.     else:
441.         Output[2] = Y[2]
442.         count += 1
443.     else:
444.         Output[1] = Y[1]
445.         count += 1
446. elif K == 5:
447.     if np.array_equal(Output[1], Y[1]) == True:
448.         if np.array_equal(Output[2], Y[2]) == True:
449.             if np.array_equal(Output[3], Y[3]) == True:
450.                 if np.array_equal(Output[4], Y[4]) == True:
451.                     if np.array_equal(Output[5], Y[5]) == True:
452.                         if (Output[1] == Y[1]).all() == True:
453.                             if (Output[2] == Y[2]).all() ==
True:
454.                                 if (Output[3] == Y[3]).all() ==
True:
455.                                     if (Output[4] ==
Y[4]).all() == True:
456.                                         if (Output[5] ==
Y[5]).all() == True:
457.                                             print("Operation
Done with : ", count, " Iteration")
458.                                             break
459.                                     else:
460.                                         Output = Y
461.                                         count += 1
462.                                 else:
463.                                     Output = Y
464.                                     count += 1
465.                             else:
466.                                 Output = Y
467.                                 count += 1
468.                         else:
469.                             Output = Y
470.                             count += 1
471.                     else:
472.                         Output[5] = Y[5]
473.                         count += 1
474.                 else:
475.                     Output[4] = Y[4]
476.                     count += 1
477.             else:
478.                 Output[3] = Y[3]
479.                 count += 1
480.         else:
481.             Output[2] = Y[2]
482.             count += 1
483.     else:
484.         Output[1] = Y[1]
485.         count += 1
486. elif K > 5:

```

```

487.         print("Process canceled, the maximum number of K is 5!!!")
488.         Break
489. # add new color and labels according to number of K
490. color=['red','blue','magenta','cyan']
491. labels=['cluster1','cluster2','cluster3','cluster4']
492. for k in range(K):
493.     plt.scatter(Output[k+1][:,1],Output[k+1][:,2],c=color[k],label=labels[k]
494.                )
495. plt.scatter(Centroids[1,:],Centroids[2,:],s=200,c='yellow',label='Centroids')
496. plt.xlabel('weight')
497. plt.ylabel('Recommended')
498. plt.title("Plot of Clustered Data")
499. plt.legend()
500. df_clust1_rec = pd.DataFrame(data = Output[1][0:,1:], index =
501.                               Output[1][0:,0].astype(int), columns = ['Weight','Recommendation'])
502. df_clust2_rec = pd.DataFrame(data = Output[2][0:,1:], index =
503.                               Output[2][0:,0].astype(int), columns = ['Weight','Recommendation'])
504. actual = pd.DataFrame(data = x[0:,1:], index = x[0:,0].astype(int),
505.                        columns = ['Weight','Recommendation'])
506. TP = 0
507. FP = 0
508. for i in range(len(df_rec)):
509.     try:
510.         if actual['Recommendation'][i] == 1:
511.             if df_clust1_rec['Weight'][i] == actual['Weight'][i]:
512.                 TP += 1
513.         else:
514.             if df_clust1_rec['Weight'][i] == actual['Weight'][i]:
515.                 FP += 1
516.     except KeyError:
517.         Continue
518. TN = 0
519. FN = 0
520. for i in range(len(df_rec)):
521.     try:
522.         if actual['Recommendation'][i] == 0:
523.             if df_clust2_rec['Weight'][i] == actual['Weight'][i]:
524.                 TN += 1
525.         else:
526.             if df_clust2_rec['Weight'][i] == actual['Weight'][i]:
527.                 FN += 1
528.     except KeyError:
529.         Continue
530.
531. jumlah_data = len(actual)
532. accuracy = (TP+TN) / jumlah_data
533.
534. print("TP :",TP)
535. print("TN :",TN)
536. print("FP :",FP)
537. print("FN :",FN)
538. print("Data :", len(x))
539. print("Akurasi :",accuracy*100,'%')

```



**0.44%** PLAGIARISM  
APPROXIMATELY

## Report #13370413

Introduction Background Currently, games are one of the easiest entertainments to find, because there are already many game sales platforms available, such as Steam Store, Epic Game Store, Blizzard, PlayStation Store and others. Not only do they sell games digitally they also provide features to download and play them directly. The Steam Store also provides a means for the community to discuss, profile to show off the number of games and playtime, chat features, and exchange and sell digital items or trade items [1]. With so many features, steam is able to invite many users, even in 2020 Steam's active users reached 120 million monthly active users with peak concurrent users reaching 24.8 million users [2]. On Steam game store page, there are features that can help users such as overall user reviews, customer reviews, and tags that have a function to navigate user. User reviews and customer reviews which are important for evaluating game quality [4]. On Steam Store, every review is sorted based on the number