Google playstore app



SECTION 1

I got the Google Play Store dataset which had the attributes App, Category, Rating, Reviews, Size, Installs, Type, Price, Content rating, Genres, Last updated, Current ver, Android ver. I found quite a deep breath of work on this dataset and therefore was intrigued to use that as my inspiration.

A	В	с	D	E	F G
1 Category	Rating	Reviews	Туре	Price	Install range
2 MEDICAL	4.19	6	0	0	0-1000
B FAMILY	4.19	2	0	0	0-1000
FAMILY	3.5	38824	0	0	100000-500000
5 GAME	4.5	5849	0	0	50000-500000
6 GAME	4.6	102107	0	0	100000-500000
7 FAMILY	4	29708	0	0	100000-500000
TRAVEL_ANI	4.4	17878	0	0	100000-500000
MEDICAL	4.3	214	1	2.99	5000-10000
0 FAMILY	4.1	321	0	0	50000-500000
1 DATING	4.1	11633	0	0	50000-500000
2 SOCIAL	3.6	58	0	0	5000-10000
3 GAME	4.7	8038	0	0	50000-500000
4 TOOLS	4.7	11018	0	0	50000-500000
5 MEDICAL	3	2	0	0	0-1000
6 FINANCE	5	12	0	0	0-1000
7 BUSINESS	4	624	0	0	50000-500000
8 COMMUNIC	3.3	78	0	0	5000-10000
9 GAME	3.8	35572	0	0	100000-500000
0 FINANCE	4.19	2	0	0	0-1000
1 FAMILY	4.4	12	0	0	0-1000
2 PERSONALIZ	3.2	114	0	0	5000-10000
3 TOOLS	4.5	60571	0	0	100000-500000
4 DATING	3.4	5	0	0	0-1000
5 FAMILY	5	2	0	0	0-1000
6 MEDICAL	4.7	11	1	15.99	0-1000
7 BUSINESS	3.9	45964	0	0	100000-500000
8 TOOLS	4.3	2158	0	0	50000-500000
9 PERSONALIZ	4.2	18280	0	0	100000-500000
0 TOOLS	4.4	55	0	0	5000-10000
1 LIBRARIES_A	4.19	2221	0	0	50000-500000
2 DATING	2.5	5377	0	0	50000-500000
3 DATING	4.1	825	0	0	50000-500000
4 FAMILY	4.1	265	1	2.99	5000-10000
5 FAMILY	10	28	0	0	0-1000

Aim:

1. To predict the number of installs for each category based on ratings, reviews and price

The following are the two precedents that used the same dataset:

1. https://www.kaggle.com/shikhabains/google-reviews-prediction/notebook

2. <u>https://www.kaggle.com/lava18/all-that-you-need-to-know-about-the-android-market</u> (The second analysis gives a good understanding of how I can approach this data going forward.)

SECTION 2

Raw features provided: App, Category, Rating, Reviews, Size, Installs, Type, Price, Content rating, Genres, Last updated, Current ver, Android ver

Features kept: Category, Rating, Reviews, Installs, Type, Price

Features modified: Installs to Number of installs range, 'Type' to numerical from a nominal attribute.

The following pre-formatting was performed on the data:

- 1. **Size:** Initially converted all app sizes into MB. After performing basic analysis, I found out that it wasn't contributing much to the aim of my analysis and therefore deleted the 'Size' attribute.
- 2. **Number of installs:** Took out '+' from the number of installs' values. Further, since the range of the number of installs was really big i.e. 0-5000k, I took ranges of the number of installs such as 0-1000 and so on and assigned instances to each based on that in such a way that each bucket contained almost the same amount of instances. This was done to ease out the analysis process such that it takes up less computational power and hopefully get more accurate results.
- 3. I deleted some of the repetitive attributes and the ones that didn't contribute a lot to my aim for this project.
- 4. **Type:** Converted this attribute to numerical i.e. 0 and 1 from a nominal one which had 'TRUE' and 'FALSE' as its class values.

Class value chosen for prediction: Number of Installs

This data was gathered directly from the Google Play Store.

Initial data exploration:

Correlation matrix

	Α	В	С	D	E
1		Rating	Reviews	Туре	Price
2	Rating	1			
3	Reviews	0.17046952	1		
4	Туре	0.03843579	-0.0861424	1	
5	Price	-0.007881	-0.0229944	0.2281156	1

SECTION 3

1. Basic Error Data Analysis:

When I saw the dataset, I knew I had to re-format the 'installs' attribute and therefore my first step was to normalize the attribute values using the formula [(Value- Min)/(Max-Min)]. As observed in the Visualize section, it immediately showed that it was still creating problems.

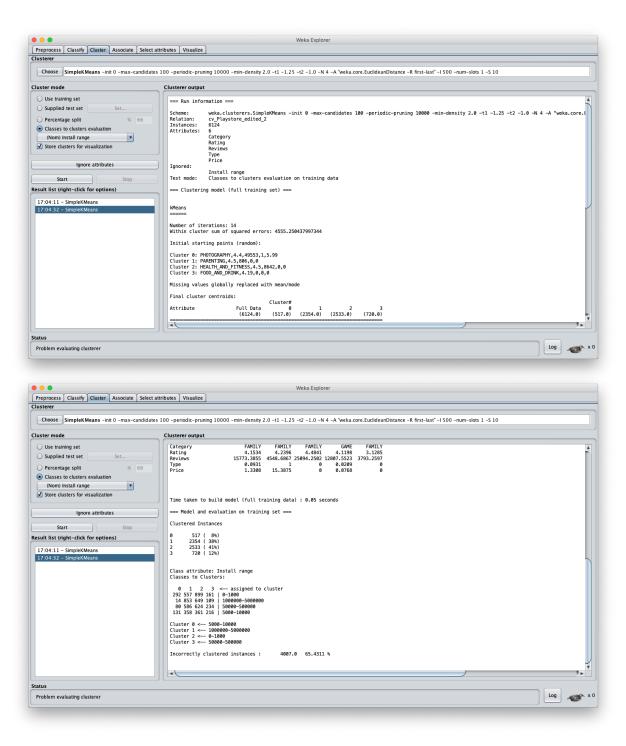
•••									Weka E	xplorer		
Preprocess CI	lassify Clus	ter Assoc	iate Selec	t attributes	Visualize							
Plot Matrix Installs	Category	Rating	Reviews	Size	Туре	Price	Content Ra	tingGenres	Last Update	d Installs	Арр	
			* **		:			-		e ^o		
Last Updated												
Genres						• • • • • • • • • •		/ [*]				
Content Rating					• • • • •	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	· · · · ·			0 000 0 000 0 000 0		
Price						Mar and a	•					
Туре	· · · · · · · · · · · · · · · · · · ·	• ••••••••••	•••		•	• • • • • •	• • • •			•••		
Size		1										
Reviews						• • •				al		
Rating											777	
Category			Riv.					<u> </u>				
POINTS178-151		()								Lindate		
Status OK												Log 🚙

•••	Weka Explorer
Preprocess Classify Cluster Associate	Select attributes Visualize
Classifier	
Choose LWL -U 0 -K -1 -A "weka.core	neighboursearch.LinearNNSearch -A \`weka.core.EuclideanDistance -R first-last\`" -W weka.classifiers.trees.DecisionStump
Test options	Classifier output
Use training set	Attributes: 11
O Supplied test set Set	Category
	Rating Reviews
Cross-validation Folds 10	Size
O Percentage split % 66	Туре
	Price
More options	Content Rating Genres
J	Last Updated
	Installs
(Num) Installs	App
	Test mode: evaluate on training data
Start Stop	=== Classifier model (full training set) ===
Result list (right-click for options)	
	Locally weighted learning
14:34:51 - misc.InputMappedClassifier	Using classifier: weka.classifiers.trees.DecisionStump
14:36:13 - misc.InputMappedClassifier	Using Linear weighting kernels
14:37:06 - misc.InputMappedClassifier	Using all neighbours
14:37:27 - misc.InputMappedClassifier	
14:38:09 - misc.InputMappedClassifier	Time taken to build model: 0 seconds
14:39:06 - misc.InputMappedClassifier	=== Evaluation on training set ===
14:39:21 - misc.InputMappedClassifier	
14:40:11 - misc.InputMappedClassifier	Time taken to test model on training data: 54.4 seconds
14:40:30 - misc.InputMappedClassifier	
17:48:41 - functions.Logistic	=== Sumary ===
17:52:35 - rules.ZeroR	Correlation coefficient 0.966
17:53:07 - functions.LinearRegression	Mean absolute error 285464.7198
18:00:39 - lazy.LWL	Root mean squared error 357321.603
	Relative absolute error 32,8821 % Root relative squared error 25,6643 %
	Total Womber of Instances 8747
Status	
ОК	Log 🗸 x0
	, <u> </u>

I wasn't able to perform even simple linear regression on this data with the class value as 'installs' as the range was too big and therefore I tried to analyze the data using Lazy weight learners which gave me an RRSE of 25%.

Clustering

I also decided to cluster my initial data to get a sense of how categories get clustered together to understand certain patterns in the dataset.



Using the classes to clusters evaluation where class value is 'Install range', it can be seen that the majority of Photography as a category with a paid type and above 4.4 rating belong to Cluster 0. The Cluster 1 majority contains Parenting, Cluster 2 contains Health and fitness and finally Cluster 3 majorly contains Food and drink. All contain majorly Free apps i.e. 0 and have a rating above 4. This categorization is not completely reliable as there are about 65% incorrectly classified instances.

2. Problematic Features: 'Installs' attribute as it had a huge range 0-5000K and the 'Size' of the app' as its class values were in different units. Another problem encountered was that the attribute 'Genres' was very similar to 'Category' but just a little more detailed and therefore I found it being repetitive and not really contributing to the analysis but just taking more computational power.

3. Ideas for improvement:

- For Installs: I initially normalized the attribute values which didn't work as it was giving me an even larger Root relative squared error (RRSE) value which is bad. I finally decided to make the attribute values nominal i.e. bucketed class values such that distribution of instances was almost same. This helped simplify the analysis and reduce the amount of computational power required.

- For Size: I didn't see too much contribution of size of the app towards the aim of my analysis.

- **Other improvement:** Removed multiple attributes such as Last updated, Current ver, Android ver and Content rating to make the analysis process more streamlined.

- Tests: After reconfiguring my data, I tried the following analysis:
- 1. Logistic regression on cv set I Accuracy: 82.0967% I RRSE: 58.7481%

			Weka E	xplorer			
Preprocess Classify Cluster Associa	te Select attributes Visualize						
Classifier							
Choose Logistic -R 1.0E-8 -M -1 -	num-decimal-places 4						
Logistic R 1.02 0 M 1	num decinar places 4						
Test options	Classifier output						
Use training set	Category=DHIING Category=EVENTS Category=FOOD AND DRINK	5.8852 1.2744	4.2003 0.6587 0.2422	2.2/40 1.0927 0.8182			A
Cross-validation Folds 10	Category=HEALTH_AND_FITNESS Category=HOUSE_AND_HOME	12.7926 0.4388	0.6947 0.1742	1.4402 0.5156			
O Percentage split % 66	Category=SPORTS Category=TOOLS Category=PHOTOGRAPHY	26.3216 2.0305 1.2266	2.9811 0.5642 0.2282	1.2247 0.6988 0.3379			
More options	Category=HOPPING Category=LIBRARIES_AND_DEMO Category=MAPS_AND_NAVIGATION	3.8716 0.9433 4.191	0.6288 0.8141 1.0371	1.0025 0.9961 0.5953			
(Nom) Install range	Category=PARENTING Category=VIDEO_PLAYERS Category=WEATHER	0.0723 1.6864 6.207	0.2121 0.8716 1.8146	0.3989 0.273 0.6169			
Start Stop Result list (right-click for options)	Category=BEAUTY Category=COMICS	0.6153	0.6861	1.1817			
14:33:35 - functions.Logistic	Category=EDUCATION Category=ENTERTAINMENT Rating Reviews Type Price	0.0961 0 3.9649 0.958 524108.8565 2.5596	0.1853 0.1657 2.2713 0.9963 13116.5163 2.5582	0.5303 0.343 1.6701 0.9998 58.0202 2.5729			
	Time taken to build model: 6.14 === Stratified cross-validation === Summary ===						
	Correctly Classified Instances Incorrectly Classified Instance Kappa statistic Mean absolute error Root mean squared error Relative absolute error Root relative squared error Total Number of Instances	7181 1566 0.757 0.130 0.252 35.218 58.748 8747	17.5 4 8 3 %	1967 % 1033 %			J
	=== Detailed Accuracy By Class						1
Status							
ок						Log	√ ×0

2. JRip rule on cv set I Accuracy: 83.8345% I RRSE: 59.5234%

•••	Weka Explorer
Preprocess Classify Cluster Associate	Select attributes Visualize
Classifier	
Choose J48 -C 0.25 -M 2	
Test options 0	Classifier output
Vet sport Vet sport	Initial Conjunt Providers - 200/1 and 11/pr - 11/set - 11/set - 11/pr - 200000 (2004/1/3/a) Initial Transmet-1880 (2227.0/236.0) Number of Fulles : 13 Initial Transmet-1880 (2227.0/236.0) Number of Fulles : 13 Tate taken to build model: 0.66 seconds ====================================
Status	
ок	

3. J48 decision trees on cv set | Accuracy: 84.2913% | RRSE: 57.9563%

0 0	Weka Explorer
	ate Select attributes Visualize
lassifier	
Choose J48 -C 0.25 -M 2	
est options	Classifier output
 Use training set 	Reviews > 41867: 1000008-5000000 (17.0/6.0)
Supplied test set Set	Number of Leaves : 150
Cross-validation Folds 10	Size of the tree : 206
Percentage split % 66	
More options	Time taken to build model: 0.11 seconds
more options	=== Stratified cross-validation ===
	== Stanary ==
om) Install range	Correctly Classified Instances 5162 84,2913 %
Start Stop	Incorrectly Classified Instances 962 15.7087 %
ult list (right-click for options)	Kappa statistic 0.7878 Mean absolute error 0.1175
	Root mean squared error 8.2493 Relative absolute error 31.755 3
13:52:00 – trees.RandomForest 13:52:27 – lazy.LWL	Root relative squared error 57.9563 %
13:53:27 - functions.SMO	Total Number of Instances 6124
14:24:28 - trees.J48	=== Detailed Accuracy By Class ===
14:44:28 - trees.J48	TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
	0.907 0.037 0.917 0.907 0.912 0.873 0.976 0.921 0-1000
	0.890 0.833 0.833 0.893 0.891 0.852 0.975 0.917 1000000-500000 0.866 0.073 0.786 0.805 0.796 0.727 0.931 0.774 5000-500000
	0.769 0.059 0.718 0.709 0.714 0.654 0.915 0.668 5000-10000
	Weighted Avg. 0.843 0.850 0.843 0.843 0.843 0.793 0.954 0.839
	=== Confusion Matrix ===
	a b c d ← classified as
	1731 0 1 177 a = 0-1000 11446 175 3 b = 1000000-5000000
	4 174 1229 117 c = 50000-500000
	151 0 159 756 j d = 5000-10000
tus	
ж	

4. Random forests on cv set I Accuracy: 82.9197% I RRSE: 57.6768%

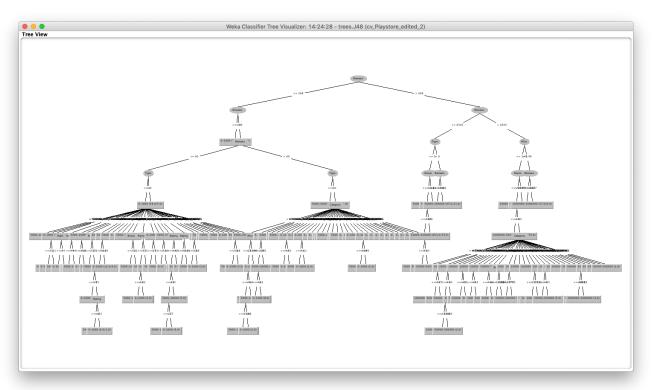
• • •	Weka Explorer
Preprocess Classify Cluster Associate	Select attributes Visualize
Classifier	
Choose SMO -C 1.0 -L 0.001 -P 1.0E-	12 - N 0 - V - 1 - W 1 - K 'weka.classifiers.functions.supportVector.PolyKernel - E 1.0 - C 250007' - calibrator 'weka.classifiers.functions.Logistic - R 1.0E-8 - M - 1 - num-decimal-places 4"
Test options	Classifier output
Ores-valdation Folds 10 Soar 3000 Result list (right-click for appleos) 13 20 27 - Pary UN 13 23 27 - Pary UN 13 23 27 - Pary UN	Randomforest bagging with 100 iterstions and base learner weaks_clossifiers.trees.handboffree +C = 1.00 + 0.001 - 5 1 - do-not-check-capabilities The taken to build socie +D.21 eccods == StratyTes ==
Status Building model for fold 8	

5. LWL on cv set | Accuracy: 57.838% | RRSE: 83.3016%

• • •	Weka Explorer
Preprocess Classify Cluster Associate	Select attributes Visualize
Classifier	
Choose SMO -C 1.0 -L 0.001 -P 1.0E-1	2 - N 0 - V - 1 - W 1 - K 'weka.classifiers.functions.supportVector.PolyKernel - E 1.0 - C 250007* - calibrator 'weka.classifiers.functions.Logistic - R 1.0E-8 - M - 1 - num-decimal-places 4*
Test options	Classifier output
 Use training set 	Locally weighted learning
O Supplied test set Set	Using classifier: weka.classifiers.trees.DecisionStump
Cross-validation Folds 10	Using linear weighting kernels Using all neighbours
O Percentage split % 66	Time taken to build model: 0 seconds
More options	=== Stratified cross-validation ===
	sees Sumary see
(Nom) Install range	Correctly Classified Instances 3542 57.838 % Incorrectly Classified Instances 2582 42.162 %
Start Select the attribute to use	as the class (the error 0, 25%)
Result list (right-click for options)	Root mean squared error 0.3583 Relative absolute error 69,5693 %
13:52:00 - trees.RandomForest	Root relative squared error 83.3016 % Total Number of Instances 6124
13:52:27 - lazy.LWL 13:53:27 - functions.SMO	
	=== Detailed Accuracy By Class ===
	TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PCL Area Class 0.996 0.220 0.672 0.996 0.803 0.721 0.920 0.775 01000
	0.997 0.363 0.498 0.997 0.664 0.561 0.876 0.686 1000000-5000000 0.014 0.005 0.500 0.014 0.027 0.048 0.813 0.515 5000-500000
	0.000 0.000 7 0.000 7 7 0.752 0.342 5000-10000 Weighted Avg. 0.578 0.166 7 0.578 7 0.653 0.610
	=== Confusion Matrix ===
	a b c d < classified as
	1991 7 1 0 a = 0-1000 3 1620 2 0 b = 100000-5000000
	109 1394 21 0 c = 50000-580000 514 234 18 0 d = 5000-10000
Status	
Building model for fold 9	Log 🦤 🗴
	, <u> </u>

As J48 turned out be the best performing algorithm on my cross-validation dataset with highest accuracy rate and lowest RRSE comparatively, I decided to go ahead with this. The RRSE is still very high which means that the 15% of the incorrectly classified instances will be quite far away from the actual instance values.

Reason: I think the reason J48 worked the best in this scenario was because my class value is to predict the number of installs based on the categories, number of ratings and reviews, type and price which can be understood very well based on setting rules and taking a top-down approach whereby the decision tree is first splitting the training data through the number of reviews as the node (<=268, >268) and then further dividing based on type and price. Further, it again divides using the number of reviews and then finally by category.



- After looking at this distribution, I thought it might be a good idea to remove the attribute of 'Type' to see if the distribution gets more simplified and maybe provide for a better accuracy but it was a little less so I decided not to go with that idea.

Training the model using J48 decision tree algorithm on cv set I Accuracy: 84.2913%

•••	Weka Explorer
Preprocess Classify Cluster Associate	Select attributes Visualize
Classifier	
Choose J48 -C 0.25 -M 2	
Test options	Classifier output
O Use training set	NEXTERP > 41001; T000000-2000000 (11.0/0.0)
Supplied test set Set	Number of Leaves : 150
Cross-validation Folds 10	Size of the tree : 206
Percentage split % 66 More options (Nom) Install range	Time taken to build model: 0.03 seconds === Stratified cross-validation === == Summary ===
Start Stop Result list (right-click for options) 14:33:35 - functions.logistic 14:35:30 - lazy.LWL 14:39:58 - rules.JRip	Correctly Classified Instances 5162 84.2913 % Incorrectly Classified Instances 962 15787 % Mean absolute error 8,7878 982 Mean absolute error 8,1788 982 Real absolute error 31,7358 % 982 Root relative garend error 37,555 % 974 Total Number of Instances 6124 974
14-02 0 - ruiss.OreR 14-033 - ues.J48 15:05:57 - ues.J48 15:05:22 - misc.inputMappedClassifier	== Detailed Accuracy by (Uss === 0.407 0.403 0.437 0.432 0.437 0.432 0.437 0.452 0.473 0.476 0.521 0.4080 0.407 0.437 0.437 0.437 0.452 0.473 0.475 0.475 0.471 0.40800 0.458 0.
Status	
ОК	

Testing the model using J48 decision tree algorithm on dev set I Accuracy: 87%

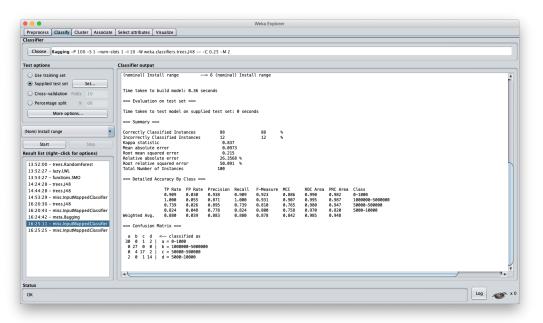
Choose J48 - C 0.25 - M2 Text options Classifier output Ubstanting set	• • •	Weka Explorer
Choose j48 - C 0.2 - M2 Test options Out staining set Supplied test set: Supp		Select attributes Visualize
Test options Classifier output Ube training set Indext // Fride Sopplied test set Sot Ordss-wildlawd Sot Ordss-wildlawd Sot Ordss-wildlawd Sot Inter taken to build model: 8.64 seconds = Evaluation on test set == Tame taken to test model on supplied test set: 0.61 seconds = Sum register Sopplied test set Sopplied test set Sopplied test set State Sopplied test set Sopplied test set Sopplied test set So	Classifier	
Uber taring set Soppliefters set <td< th=""><th>Choose J48 -C 0.25 -M 2</th><th></th></td<>	Choose J48 -C 0.25 -M 2	
Uber uning set Supplied trass Supplied trass set Supplied trass set New processing spik is 500 Percentage spik is 510 Percentage spik is	Test options	Classifier output
Image Summary Same Summary Same Summary Incorrectly (Classified Instances 17 1423353 - functions.Logiski Same and Solute error 1423353 - functions.Logiski 0 142335 - functions.Logiski The take to the function of Instances 17 142335 - functions.Logiski 10 142335 - functions.Logiski 10 142335 - functions.Logiski 10 142435 - functions.logiski 10 150557 - functions 10 150557 - functions 10 142435 - functions 10 150557 - functions 0.000 0.000 0.000 12004 0.000 0.000 0.000 0.000 0.000 1200522 - muclicuput/dippedClastef 0.000 0.000 <	Supplied test set Set	(nominal) Install range → 6 (nominal) Install range
Nominal ange Image: Constraint of participation Start Star Start Start		
15:05:37 - tres:/48 16:05:38 16:05:38 16:05:38 16:05:38 17:05:38 18:05:38 18:05:38 19:06:38 19:07:38 19:08:38 19:08:38 19:08:38 <	Start Stop Result list (right-ctick for options) 14:33:35 - functions.Logistic 14:33:33 - fazy.LWL 14:35:30 - lazy.LWL 14:39:58 - rules,Rkp 14:40:20 - nules,ORR	Correctly Classified Instances 87 87 % Incorrectly Classified Instances 12 13 % Wean absolute error 8.0906 Root maan squared error 8.2233 Rolative absolute error 9.2235 % Noor relative squared error 52.0227 % Troit Namer or Instances 100
	15:05:57 - trees.J48	8.999 0.045 0.999 0.949 0.949 0.949 0.950 0.951 0.952 0-1040 10.000 0.055 0.952 0.959 0.952 0.952 0.952 0.951 100000-000000 0.695 0.000 1.000 0.695 0.955 0.952 0.955 0.955 0.955 0.955 0.955 0.955 0.950 0.955 0.955 0.950 0.955 0.955 0.950 0.955 0.955 0.955 0.955 0.95 0
OK Log ស X 0	Status	
	ОК	Log 🛷 x

4. Evaluation:

• • •	Weka Explorer
Preprocess Classify Cluster Associate	Select attributes Visualize
lassifier	
Choose Bagging -P 100 -S 1 -num-sl	ts 1 - I 10 -W weka.classifiers.trees.J48 C 0.25 -M 2
est options	Classifier output
 Use training set Supplied test set SetL Operating a set of the se	<pre>=</pre>
tatus	
OK	
UN	······································

A. Based on the J48 results on the dev set, I also tried bagging to train the data for better results and then test with the dev set.

The accuracy improved a little to **84.3876%** and on the dev set the accuracy was **88%** which was really good.



B. Finally, I performed tuning in the next section on the best performing model to see if I can improve the accuracy of the model by tinkering the minNumObj.

SECTION 4

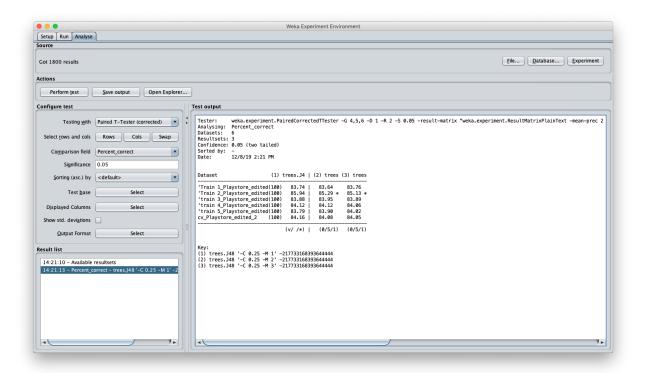
Based on my initial experimentation with multiple algorithms on the complete crossvalidation set and further using that training results for prediction evaluation on my dev set, I decided to take the best performing algorithm which was the J48 decision tree algorithm on my dataset for my tuning process for classification. I decided to tune the minNumObj parameter. Range tuned is 1-3.

Fold	Train set performa nce for exponent = 1	Train set performa nce for exponent = 2	Train set performa nce for exponent = 3	Optimal Setting	Test set performa nce -OS	Default Setting	Test set performa nce - DS
1	83.74	83.64	83.76	3	84.5714	2	84.4082
2	85.94	85.29	85.13	1	83.1837	2	82.5306
3	83.88	83.95	83.89	2	84.0	2	84.0
4	84.12	84.12	84.06	1	84.898	2	84.6531
5	83.79	83.90	84.02	3	83.9052	2	83.7418
Average	84.294	84.18	84.172		84.11166		83.86674

Average of each performance based on percent correct:

Performance of exponent 1 = 84.294 Performance of exponent 2 = 84.18 Performance of exponent 3 = 84.172

By taking the difference between the Optimal and default setting on the test set performance average and performing a t-test on it, I found out that the P value is 0.08899883 which is more than 0.05 and that means that the model with the new settings shows insignificant improvement from its default settings i.e. 2 as the minNumObj; therefore tuning was not worth it.



Paired T-test

	А	В
1	84.5714	84.4082
2	83.1837	82.5306
3	84	84
4	84.898	84.6531
5	83.9052	83.7418
6		
7	0.08899883	

SECTION 5

Step 1: Testing on the trained model using the J48 decision tree algorithm with default settings as the best found settings. Accuracy: 88.1356% I RRSE: 53.4142%

•••	Weka Explorer
Preprocess Classify Cluster Associate	Select attributes Visualize
Classifier	
Choose J48 -C 0.25 -M 2	
Test options	Classifier output
Use training set Supplied test set Supplied test set Cross-validation Folds 10 Percentage split % 66 More options (Nom) Install range Start Soop Start	Innueric) Price →> 5 (numeric) Price (nominal) Install range →> 6 (nominal) Install range Time taken to build model: 0.04 seconds === Evaluation on test set == Time taken to test model on supplied test set: 0 seconds == Summary == Correctly Classified Instances 184 Road ream squared error 0.3042 Road ream squared error 0.2287 Relative abcolute error 0.3042 Road ream squared error 0.2287 Relative abcolute error 1.3644 % 0.093 0.022 0.926 0.933 0.909 0.952 0.952 0.957 0.333 0.909
	a b c d < Classified as 40 0 0 1 a = 0-1000 0 2 3 0 b = 1000000-500000 1 2 2 2 3 c = 50000-500000 1 0 3 14 d = 5000-10000 *
Status	
ОК	Log 🐠 x0

Step 2: Testing on the trained model using the J48 decision tree algorithm with default settings as the best found settings and adding an ensemble method of bagging to it.

	Weka Explorer	
Preprocess Classify Cluster Associate	Select attributes Visualize	
Classifier		
Choose Bagging -P 100 -S 1 -num-slot	ts 1 -1 10 -W weka.classifiers.trees.J48C 0.25 -M 2	
Test options	Classifier output	
O Use training set	(nominal) Install range> 6 (nominal) Install range	
Supplied test set Set	Time taken to build model: 0.36 seconds	
Cross-validation Folds 10		
O Percentage split % 66	=== Evaluation on test set ===	
More options	Time taken to test model on supplied test set: 0 seconds	
	=== Summary ===	
(Nom) Install range	Correctly Classified Instances 102 86.4407 % Incorrectly Classified Instances 16 13.5593 %	
Start Stop	Kappa statistic 0.8146 Mean absolute error 0.1	
Result list (right-click for options)	Root mean squared error 0.2164	
13:52:00 - trees.RandomForest	Relative absolute error 27.1545 % Root relative squared error 50.5396 %	
13:52:27 - lazy.LWL	Total Number of Instances 118	
13:53:27 - functions.SMO 14:24:28 - trees.I48	=== Detailed Accuracy By Class ===	
14:44:28 - trees.J48	TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class 0.951 0.039 0.929 0.951 0.940 0.907 0.987 0.961 0-1000	
14:53:29 - misc.InputMappedClassifier 16:20:30 - trees.J48	0.857 0.011 0.960 0.857 0.906 0.881 0.992 0.978 1000000-5000000	
16:20:30 - trees.J46 16:20:43 - misc.InputMappedClassifier	0.806 0.065 0.806 0.806 0.806 0.737 0.550 0.888 50000-500000 0.778 0.066 0.700 0.778 0.588 0.954 0.51 5000-10000	
16:24:42 - meta.Bagging	Weighted Avg. 0.864 0.043 0.869 0.864 0.866 0.823 0.975 0.929	
16:26:19 - misc.InputMappedClassifier	=== Confusion Matrix ===	
	a b c d < classified as	
	0 24 4 0 b = 1000000-5000000	
	1 1 25 4 c = 50000-500000 2 0 2 14 d = 5000-10000	
		2
	•(7.6
Status		
ОК		✓ × 0

Accuracy: 86.4407% | RRSE: 50.5396%

While the accuracy increased while testing on the dev set using bagging, the accuracy decreased with the test set using bagging. I therefore decided to not consider this method in my analyzation further.

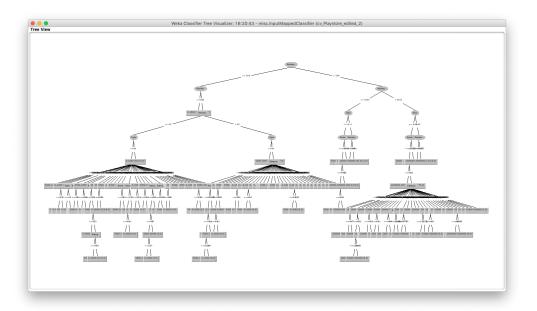
Step 3: Testing on the trained model using the J48 decision tree algorithm with default settings as the best found settings and performing cost- sensitive analysis on it to try to improve performance.

•••	Weka Explorer	
Preprocess Classify Cluster Associate	Select attributes Visualize	
Classifier		
Chapter Constituenting Interifier and	-matrix "[0.0 1.0 1.0 2.0; 1.0 0.0 2.0 1.0; 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0]" - S 1 - W weka.classifiers.trees.J48 C 0.25 -M 2	
Clouse CostSensitiveClassifier-cost	mautx [0.0 1.0 1.0 2.0, 1.0 0.0 2.0 1.0, 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0] -5 1 -W weka.taassinets.tees.j48C 0.25 -M 2	
Test options	Classifier output	
 Use training set 	(nominal) Install range> 6 (nominal) Install range	
Supplied test set Set		
Cross-validation Folds 10	Time taken to build model: 0.04 seconds	
O Percentage split % 66	=== Evaluation on test set ===	- 11
More options	Time taken to test model on supplied test set: 0 seconds	
	== Sumary ===	- 1
Alers have a second		
(Nom) Install range	Correctly Classified Instances 183 87.2881 % Incorrectly Classified Instances 15 12.7119 %	- 1
Start Stop	Kappa statistic 0.8242	- 1
Result list (right-click for options)	Mean absolute error 0.1062 Root mean squared error 0.2303	- 1
	Relative absolute error 28.8261 % Root relative source error 53.738 %	- 1
13:52:00 - trees.RandomForest	Root relative squared error 55.7835 % Total Number of Instances 118	- 1
13:52:27 - lazy.LWL 13:53:27 - functions.SMO		- 1
14:24:28 - trees.148	=== Detailed Accuracy By Class ===	- 1
14:44:28 - trees.148	TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class	- 1
14:53:29 - misc.InputMappedClassifier	1.000 0.052 0.911 1.000 0.953 0.929 0.990 0.969 0-1000	- 1
16:20:30 - trees.J48	0.893 0.022 0.926 0.893 0.909 0.882 0.978 0.927 1000000-500000 0.839 0.080 0.788 0.839 0.813 0.744 0.930 0.927 100000-500000	- 1
16:20:43 - misc.InputMappedClassifier	0.611 0.020 0.846 0.611 0.710 0.679 0.953 0.752 5000-30000	- 1
16:24:42 - meta.Bagging	Weighted Avg. 0.873 0.048 0.872 0.873 0.869 0.831 0.966 0.880	- 1
16:26:19 - misc.InputMappedClassifier	=== Confusion Matrix ===	- 1
16:37:14 - trees.J48		- 1
16:38:13 - meta.CostSensitiveClassifier	a b c d < classified as	
16:39:07 - meta.CostSensitiveClassifier	$41 \ 0 \ 0 \ 0 a = 0 - 1000$	
16:39:39 - meta.CostSensitiveClassifier	0 25 3 0 b = 100000-5000000 1 2 26 2 c = 5000-5000000	
16:40:25 - meta.CostSensitiveClassifier	30 4 11 d = 5000 - 10000	
16:42:07 - meta.CostSensitiveClassifier		
16:43:01 - misc.InputMappedClassifier		
		7.0
itatus		
ОК		
UN		æ

The performance didn't improve but actually got a little worse.

I therefore finally went with the first model without the bagging or the cost-sensitive analysis as it gave the best results on the final test set.

SECTION 6



Based on the best performing model above which has an accuracy of 88.1356%, it can be concluded that majority of the instances from a test set would be correctly predicted into one of the 4 ranges for the number of installs thereby suggesting that given a particular category, rating, number of reviews, type and price of app, it is an 88% chance that number of installs for that app category will be correctly predicted.

Through this project I learnt a lot about the benefits and disadvantages of various algorithms which helped in understanding how to build the best prediction model.