Names:	*Alex Randles (15302766), Karl Whelan (15561423)*
Programme	*CASE*
Module Code	*CA4010*
Assignment Title	*Data Warehousing & Data Mining report*
Submission Date	*17/11/2018*
Module Coordinator	*Mark Roantree*

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I /We have read and understood the Assignment Regulations. I/We have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

I/We have read and understood the referencing guidelines found at
http://www.dcu.ie/info/regulations/plagiarism.shtml
https://www4.dcu.ie/students/az/plagiarism
and/or recommended in the assignment guidelines.

Name(s): Alex Randles & Karl Whelan *Date: 16/11/1/18*

Section 1 - Idea and Dataset description

1.1 Idea

- *The idea came to us after we had read through already approved Project ideas. The one that peaked our interest was "Predict the revenue of box office movies". Since we were both avid movie watchers, we though some idea related to movies would be great. We then looked through kaggle to find datasets which were some way related to movies. We found ones related to movies profits, actors , genres and lots more. We then stumbled across a dataset which contained the IMDB scores of each movie. Since we both agreed that IMDB score was the most accurate representation of the quality of a movie, we decided to pick the project idea "Predict the IMDB score". We really wanted to know what were the main factors thats went into a movie recieving a higher IMDB score and to see if any assumptions we made about which factors contributed to a higher score we're correct.*

1.2 Dataset Description

- *The dataset we are using is a kaggle dataset found at the following link: (https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset)*
- *This dataset contains 5044 movies with their corresponding IMDB score. We felt 5044 movies was enough to make accurate enough prediction and to train a training set.*
- *The dataset contained 28 attributes for each movie, some we believe to be relevant and irrelevant to our predictions.*
- *List of Attributes*

color	*Was the movie black or white/color.*
director_name	*The name of the director of the movie.*
num_critic_for_reviews	*The number of critics that reviewed the movie.*
duration	*How long was the movie.*
director_facebook_likes	*The number of director facebook likes.*
actor_3_facebook_likes	*The number of facebook likes for the third most important actor of the movie.*
actor_2_name	*The second most important actors name.*
actor_1_facebook_likes	*The number of facebook likes for the first most important actor of the movie.*
gross	*Total earnings of that movie.*

genres	*The genres that the movie fell into. This could be a number of genres.*
actor_1_name	*The most important actor of the movie.*
movie_title	*The name of the movie. Used mainly as a key.*
num_voted_users	*The number users that voted for that movie.*
cast_total_facebook_likes	*The total number of likes for the entire cast.*
actor_3_name	*The name of the third most important actor*
facenumber_in_poster	*The number of faces in their movie poster.*
plot_keywords	*Keywords mentioned or about the movie.*
Movie_imdb_link	*The link to the movies IMDB page.*
num_user_for_reviews	*The number of users that reviewed that movie.*
language	*The main language used in that film.*
country	*Which country the movie was filmed in.*
content_rating	*The rating of the movies content. E.g PG-13*
budget	*The amount of money spent on filming the movie.*
title_year	*The year the movie was released.*
imdb_score	*The IMDB score, this is the value were trying to predict.*
aspect_ratio	*The aspect ratio of which the film was made.*
movie_facebook_likes	*Number of facebook likes the movie received.*

- *These attributes were of 2 different types: Continuous (Numerical) and Categorical attributes.*
- *Examples of Numerical Attributes:*
  - *IMDB Score*
  - *Facebook Likes*
  - *Budget*
- *Examples of Categorical Attributes:*
  - *Content Rating*
  - *Genre*
  - *Country*
- *This dataset is the highest quality one we could find that contained a large enough number of movies IMDB scores and the attributes we felt that may have the most effect on the score itself.*

Section 2 – Data Preparation

2.1 Remove empties

Our data-set initially had a lot of missing values for various attributes and many entries contained multiple missing values. Here is an example of one of those entries:

	Doug Walker			131		F

These missing values were spread out across many different attributes so we decided the best way to deal with this was to remove any entry that contained a missing value. We also removed duplicate entries. This left us with 3656 entries which we felt was still a reasonable size for our data-set

2.2 Breakdown of data

With the missing values taken care of this was the breakdown of our attributes.

#	**Attributes**	**Type of attribute**	**Details**
1	color	Categorical	**Unique values:** 2
2	director_name	Categorical	**Unique values:** 1659
3	num_critic_for_reviews	Continuous	**Range:** 2 - 813
4	duration	Continuous	**Range:** 37 - 330
5	director_facebook_likes	Continuous	**Range:** 0 - 23000
6	actor_3_facebook_likes	Continuous	**Range:** 0 - 23000
7	actor_2_name	Categorical	**Unique values:** 2188
8	actor_1_facebook_likes	Continuous	**Range:** 0 - 640000
9	gross	Continuous	**Range:** 162 - 760505847
10	genres	Categorical	**Unique values:** 745
11	actor_1_name	Categorical	**Unique values:** 1428
12	movie_title	Categorical	**Unique values:** 3655
13	num_voted_users	Continuous	**Range:** 91 - 1689764
14	cast_total_facebook_likes	Continuous	**Range:** 0 - 656730
15	actor_3_name	Categorical	**Unique values:** 2587
16	facenumber_in_poster	Continuous	**Range:** 0 - 43
17	plot_keywords	Categorical	**Unique values:** 3656
18	movie_imdb_link	Categorical	**Unique values:** 3656
19	num_user_for_reviews	Continuous	**Range:** 4 - 5060

20	*language*	*Categorical*	***Unique values:** 34*
21	*country*	*Categorical*	***Unique values:** 45*
22	*content_rating*	*Categorical*	***Unique values:** 12*
23	*budget*	*Continuous*	***Range:** 218 - 12215500000*
24	*title_year*	*Continuous*	***Range:** 1927 - 2016*
25	*actor_2_facebook_likes*	*Continuous*	***Range:** 0 - 137000*
26	*imdb_score*	*Continuous*	***Range:** 1.6 - 9.3*
27	*aspect_ratio*	*Continuous*	***Range:** 1.18 - 16.0*
28	*movie_facebook_likes*	*Continuous*	***Range:** 0 - 349000*

2. 3 Remove attributes with too many unique values

We only wanted to keep attributes that would help us with our predictions. It's clear that the attributes "movie_imdb_link", "movie_title" and "plot_keywords" are entirely unique for each entry so we dropped these attributes.

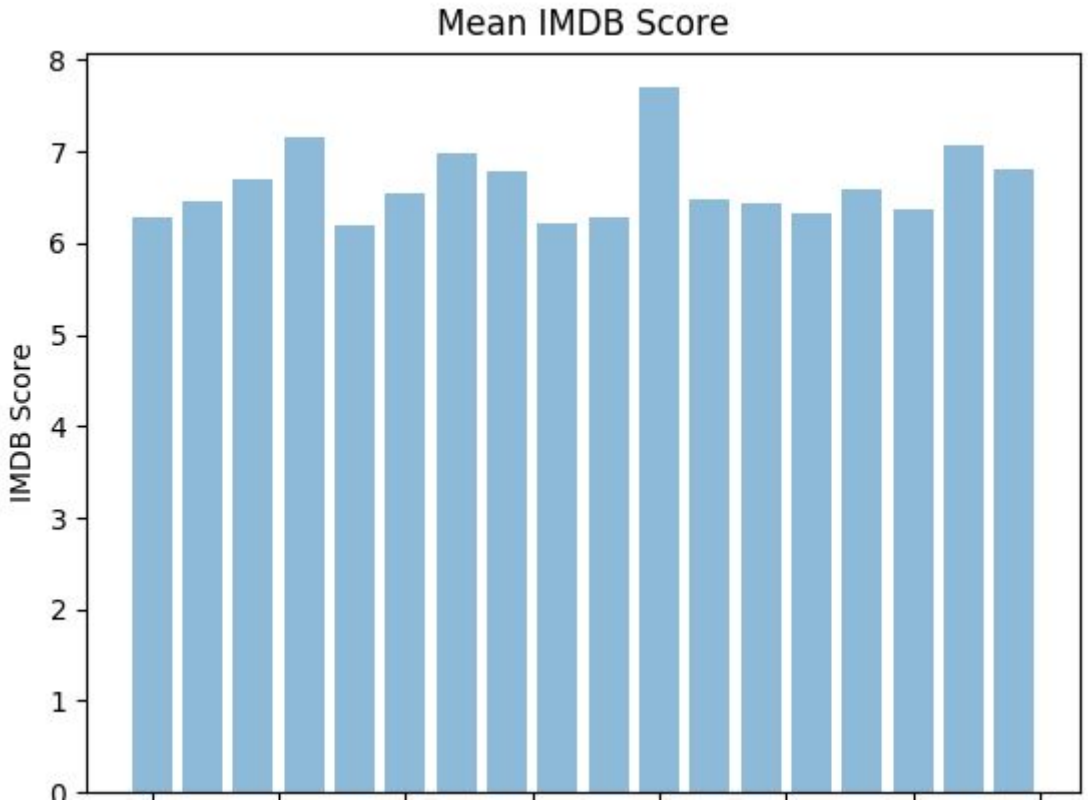
We can see from this table that several of our categorical attributes have a very large number of unique values (>1400). Specifically, all the attributes that involve names:

- *director_name: 1659*
- *actor_1_name: 1428*
- *actor_2_name: 2188*
- *actor_3_name: 2587*

As the values for these attributes will be so varied they will not be of use to help with our predictions so we dropped these attributes also.

2.4 Remove genre

Bellow you see a barchart of the mean IMDB scores for each genre. We expected that genre would have a big effect on IMDB score but as you can the average score for each genre was very similar and as such would not be helpful with predictions. Because of this we decided to drop genres as an attribute.



2.5 Remove attributes in which values occur too often

The remaining 4 categorical attributes all have a reasonable number of unique values to work with however some of the values occur so often that they are nearly constant.

Attributes	***Most occurring***	***Second most***	***Third most***
color	*Color: 3600 (96.7%)*	*Black and White: 123 (3.3%)*	*NA*
language	*English: 3566 (95.8%)*	*French: 34 (0.9%)*	*Spanish: 23 (0.6%)*
country	*USA: 2961 (79.5%)*	*UK: 313 (8.4%)*	*France: 101 (2.7%)*
content_rating	*R: 1687 (45.3%)*	*PG-13: 1291 (34.7%)*	*PG: 563 (15.1%)*

We can see from the table above that the "color" attribute will nearly always be "Color" and the language attribute will nearly always be "English". This means they will not be helpful for our prediction so we dropped these attributes.

We can also see that 79.5% of movies are made in the USA while many of the other countries only account for a small percentage. Rather than drop this attribute we decided to group all movies made outside the USA together

country	*USA: 2961 (79.5%)*	*Non USA: 762 (20.5%)*

Section 3 – Our Algorithm

After the data preparation was done our data-set looked like this

#	***Attributes***	***Type of attribute***	***Details***
1	*num_critic_for_reviews*	*Continuous*	***Range:*** *2 - 813*
2	*duration*	*Continuous*	***Range:*** *37 - 330*
3	*director_facebook_likes*	*Continuous*	***Range:*** *0 - 23000*
4	*actor_3_facebook_likes*	*Continuous*	***Range:*** *0 - 23000*
5	*actor_1_facebook_likes*	*Continuous*	***Range:*** *0 - 640000*
6	*gross*	*Continuous*	***Range:*** *162 - 760505847*
7	*num_voted_users*	*Continuous*	***Range:*** *91 - 1689764*
8	*cast_total_facebook_likes*	*Continuous*	***Range:*** *0 - 656730*
9	*facenumber_in_poster*	*Continuous*	***Range:*** *0 - 43*
10	*num_user_for_reviews*	*Continuous*	***Range:*** *4 - 5060*
11	*country*	*Categorical*	***Unique values:*** *2*
12	*content_rating*	*Categorical*	***Unique values:*** *12*
13	*budget*	*Continuous*	***Range:*** *218 - 12215500000*
14	*title_year*	*Continuous*	***Range:*** *1927 - 2016*
15	*actor_2_facebook_likes*	*Continuous*	***Range:*** *0 - 137000*
16	*imdb_score*	*Continuous*	***Range:*** *1.6 - 9.3*
17	*aspect_ratio*	*Continuous*	***Range:*** *1.18 - 16.0*
18	*movie_facebook_likes*	*Continuous*	***Range:*** *0 - 349000*

As we can see most of our attributes are continuous. For this reason we decided to use K Nearest Neighbours as our algorithm.

We Then shuffled our data-set and split it into 80% for training and 20% for testing. Our training data-set consists of 2978 entries while our testing data consists of 745 entries.

3.1 Encoding categorical values

We still had 2 categorical variables in our data-set which would not help us calculate the euclidean distance, to solve this we needed to encode them to continuous variables. We considered using one hot one encoding but we felt, given we already had a very high number of attributes(19), this would be a bad idea Calculating the euclidean distance becomes unhelpful with high dimensions as all vectors

are almost equidistant to the search query vector. Instead we used label encoding by just assigning a number to each unique value of the attributes. E.g "USA" -> 0, "Non USA" -> 1

3.2 Normalization

We then normalised our data to avoid the larger numbers skewing our euclidean distance measurement. We did this by applying the following formula, from the notes, to all the values in the data-sets.:

$$\frac{a - min}{max - min}$$

This normalised all attributes so that the values ranged from 0 to 1

3.3 The Algorithm

we then applied our algorithm.

```
def knn(train_data, test_data, k_value):
    new_test_data = []
    for i in test_data:
        euclid_dist = []
        knn = []
        knn_imdb_scores = []
        for j in train_data:
            dist = euclidean_distance(i, j)
            euclid_dist.append((j[0], dist))
            euclid_dist.sort(key=lambda x: x[1])
            knn = euclid_dist[:k_value]
            for k in knn:
                knn_imdb_scores.append(k[0])
        mean_imdb_score = sum(knn_imdb_scores) / len(knn_imdb_scores)
        new_test_data.append(numpy.append(i, mean_imdb_score))
```

It takes the training data, testing data and a number k. for every entry in the test data it calculates the k entries in the training data with the lowest euclidean distance and gets their imdb scores. It then adds the mean of those scores to a new prediction attribute in the test data. To test for accuracy the prediction is measured against the actual imdb score for each row. If the predicted score is within 0.2 of the actual score it is considered accurate. This is done for different values of k to find the optimal value.

After testing this algorithm we found the accuracy to be very low, 15% for the least optimal k to 30% for the most optimal k. We believed this was caused by the curse of dimensionality. We have 19 attributes which is throwing off our euclidean distance measurements. We decided to combat this using dimension reduction.

We found that the "cast_total_facebook_likes" had a strong correlation with "actor_1_facebook_likes" so we dropped the former attribute. we also found a strong correlation with "num_critics_for_reviews", "num_users_for_reviews" and "num_voted_users" and that "num_voted_users" had a higher correlation with "imd_score" so we dropped the other 2 attributes. Unfortunately this had very little effect on the accuracy.

We then decided that it was too ambitious to predict the exact imd_score so we categorised scores into the following:

score <= 2	*terrible*
2 < score <= 4	*bad*
4 < score <= 6	*average*
6 < score <= 8	*good*
8 < score <= 10	*very good*

After this our accuracy improved but not by as much as we'd hoped for. The final results for the accuracy for each k is in the table below. The best value of k is either 16 or 20.

k	***accuracy***
1	*42.95%*
2	*43.76%*
3	*44.43%*
4	*44.56%*
5	*43.76%*
6	*44.70%*
7	*44.70%*
8	*45.50%*
9	*44.97%*
10	*45.37%*
11	*46.31%*
12	*47.25%*
13	*47.52%*
14	*47.52%*
15	*47.79%*
16	*48.05%*
17	*47.79%*
18	*47.65%*
19	*47.65%*

20	*48.05%*

Section 4 - Results and Analysis

4.1 Dataset Analysis

- *We discovered that the dataset contained a lot of null entries and duplicates which was not good for making our predictions.*
- *The dataset also contained a lot of attributes which made it hard to find any major correlation between a certain attribute and the IMDB score.*
- *Many attributes in the dataset were eventually dropped for our predictions as they didn't influence the score as much as we would of liked.*
- *The dataset contained too many unique values which made predictions very hard, if we were to pick another dataset we would go with one that has less attributes that were more relevant and also one which has less null entry/duplicates.*

4.2 Prediction Analysis

- *While our predictions were not as accurate as we hoped, we felt that we learned a great deal about manipulating and training data which could be applied to other datasets in the future.*
- *In hopes of improving predictions we tried not to predict the exact score but a range, using this technique helped quite a bit.*
- *Dropping attributes which we felt were less relevant also helped a great deal in making our predictions.*

4.3 Results

- *Using the K nearest neighbours algorithm was the most effective for our dataset, we tried other ones which can much less accurate predictions.*
- *Classifying the IMDB scores improved accuracy.*
- *We discovered the best value for k was 16 and 20 both with a value of 48.05%. This is just shy of 50% accuracy which isn't as high as we hoped.*
- *We would most likely pick a different dataset for this if we were to do it again as we were hoping for much higher accuracy in our predictions.*