CSE 5243: Introduction to Data Mining

Analysis of Iris and Income datasets

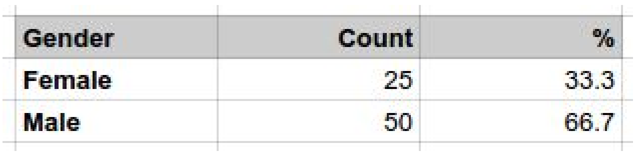
CHANDRASEKAR SWAMINATHAN (.42)

Exploratory Analysis of Income dataset

The income dataset contains 16 features namely ID, age, workclass, fnlwgt, education, education_cat, marital_status, occupation, relationship, race, gender, capital_gain, capital_loss, hour_per_week, native_country, class. The following observations can be quickly made about the dataset:

- More than 50% of the dataset covers the 20-45 age range.
- 84% white race, 16% other races (black, asian-pac-islander, amer-indian-eskimo & others)
- 65% male, more skewed towards male population
- Focusses primarily on people whose native country is USA, with nearly 90% having US as their native country
- 80% of the people earn less than 50K.
- 70% work in private sector, 14% work in the government (federal, state, local)
- 75% work 40 or lesser hours in a week.
- All the records have an unique value for the ID column.

These values are obtained by looking at the quantiles (for ratio attributes) and the histogram counts for categorical data (gender, country, etc). Having this distribution helps us get a general idea of the nature of bias/skew in the dataset. This helps us determine if certain patterns we find in the data are important or if they are just reflective of the original bias in the data.



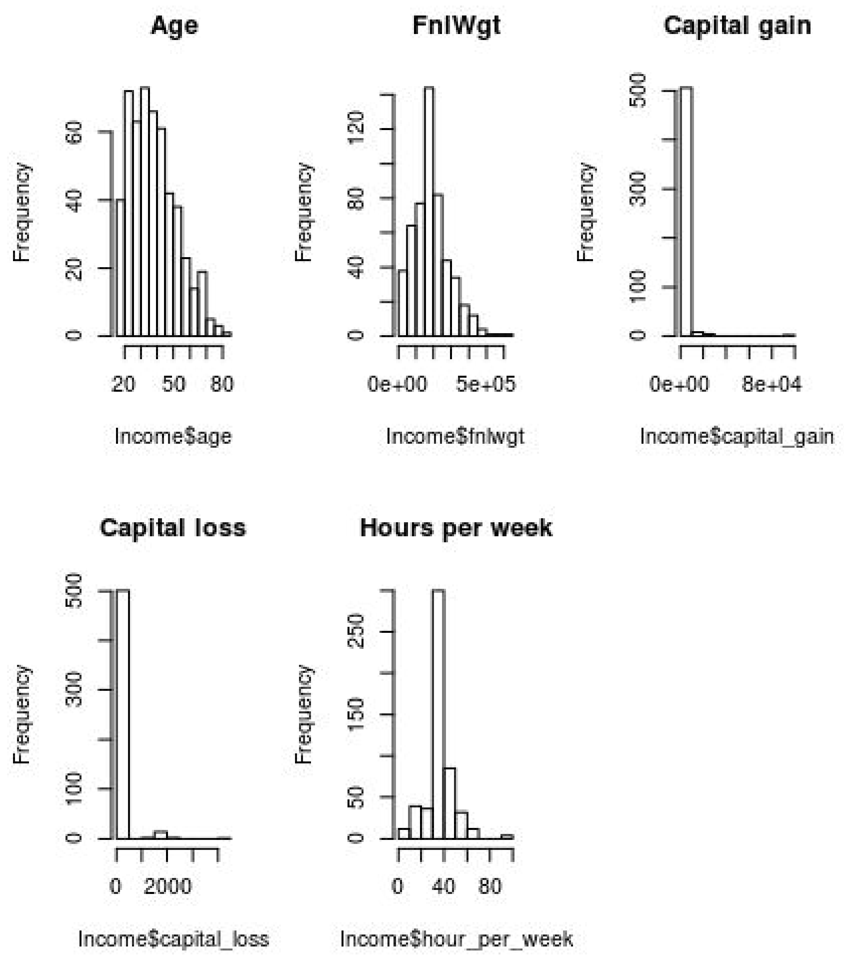
Gender	Count	%
Female	25	33.3
Male	50	66.7

For eg, if we look at the gender of people working in the government, we get the above distribution. This makes it seem like there are more male employees in the government compared to female employees. However, it is not an interesting pattern as the percentage is reflective of the overall distribution of the data.

Data distribution

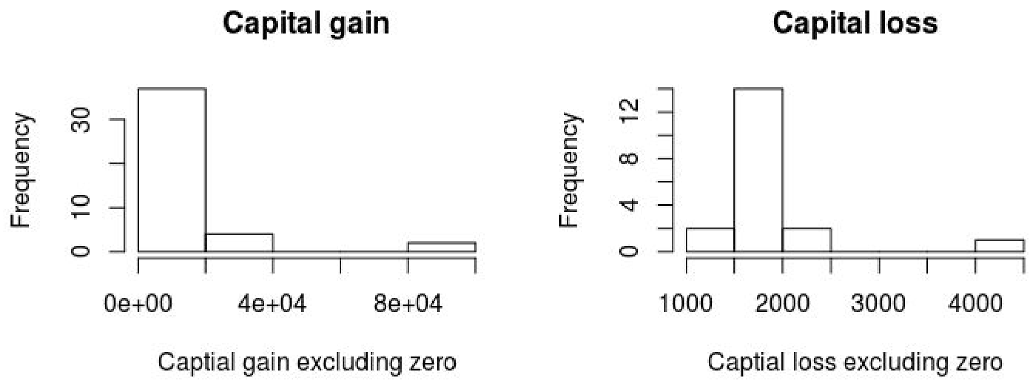
For ratio attributes, the overall distribution of the values was used to determine if any type of transformation of the data is necessary. The histogram of all the ratio features age, captial_gain, captial_loss, hours_of_week and fnl_wgt were plotted and can be seen in ***Figure 1***.

Figure 1



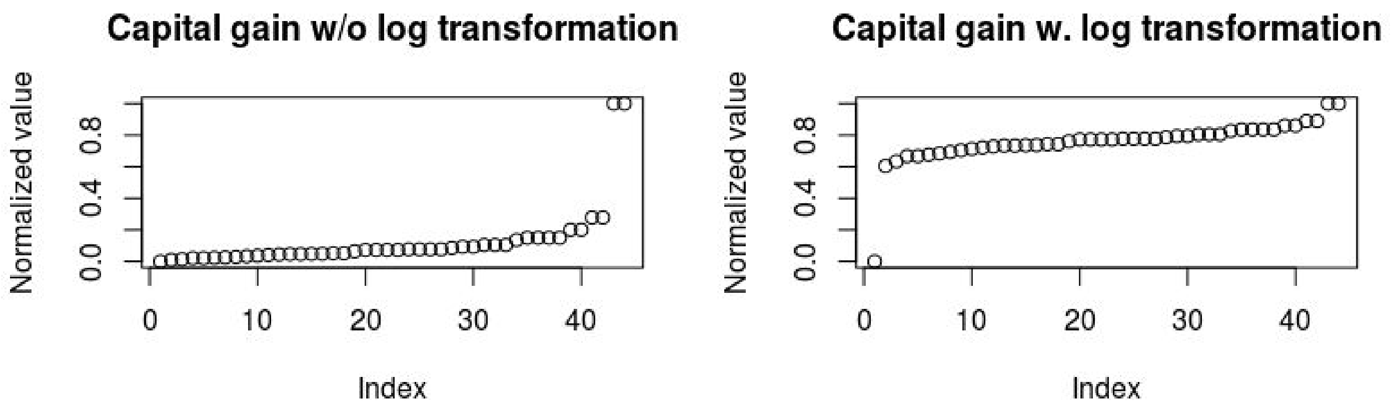
We can see that Age, FnlWgt and Hours per week are unimodal and are not heavily skewed. Even though capital loss and capital gain look like they are skewed left, it is primarily due to the zero value and looking at the distribution excluding zero values gives us a better picture. The histogram without zero value is shown in ***Figure 2***.

Figure 2



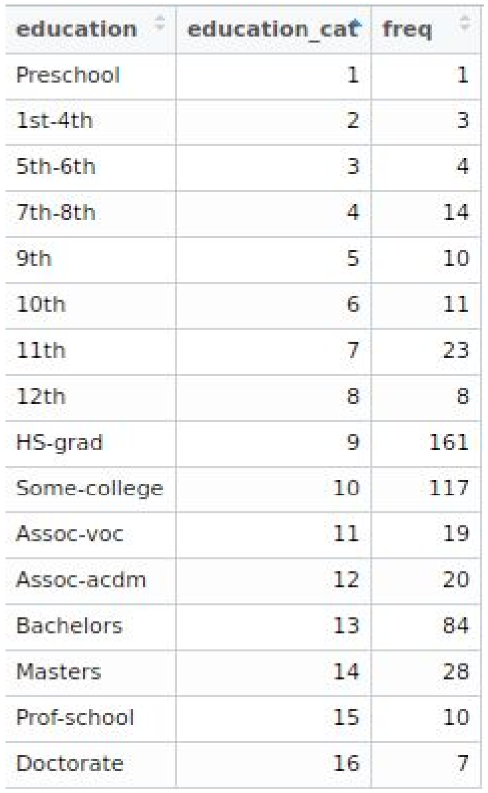
Now we can see that the capital gain feature still skewed left, whereas the capital loss feature seems to be unimodal. Normalizing the capital gain feature between 0 and 1 would lead to extremely low values being assigned to values near the head of the distribution. We have to apply a log transformation to this feature to ensure that it normalized proportionally. ***Figure 3*** shows the normalized values for capital gain with and without log transformation.

Figure 3



Interesting patterns

The following interesting patterns could be found by looking at multiple features in the data together. We can see that the **education_cat** feature is a numerical mapping for the **education** categorical feature. We can determine this by doing a group by based on both **education_cat** and **education**. The result of the group by operation is shown here.

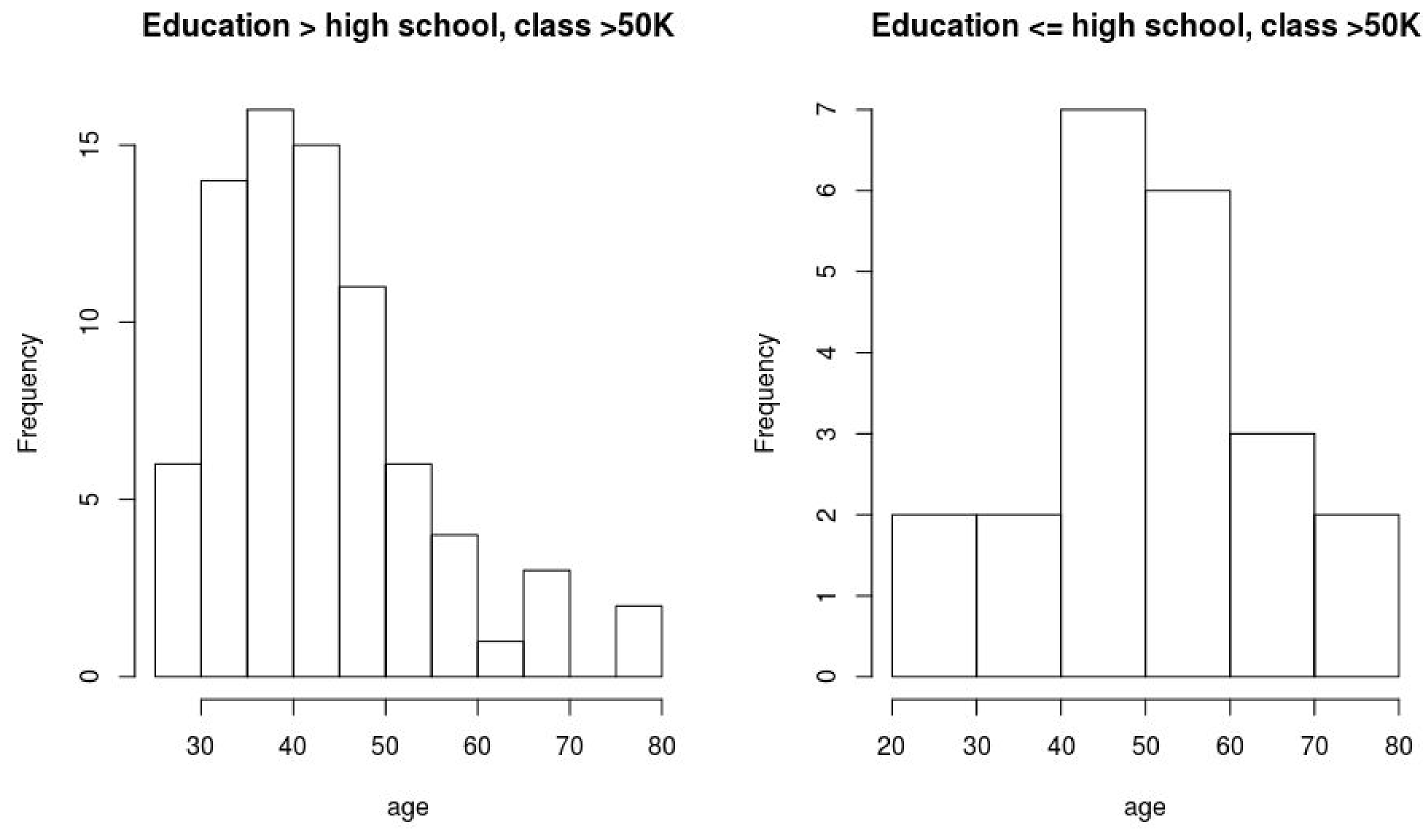


education	education_cat	freq
Preschool	1	1
1st-4th	2	3
5th-6th	3	4
7th-8th	4	14
9th	5	10
10th	6	11
11th	7	23
12th	8	8
HS-grad	9	161
Some-college	10	117
Assoc-voc	11	19
Assoc-acdm	12	20
Bachelors	13	84
Masters	14	28
Prof-school	15	10
Doctorate	16	7

We can see that each of the education_cat and education feature values appear in exactly one group. This indicates that there is a one to one mapping between both of them. Looking at the values for both these features, we can see that as the education_cat value increases, the level education indicated by the education feature is higher. For eg. preschool education level has the minimum value in education_cat, this represents the lowest level of education possible. Doctorate represents the highest level of education a person can receive and this has the highest value in the education_cat column. From this we can conclude that education_cat is an ordinal value mapping for education column.

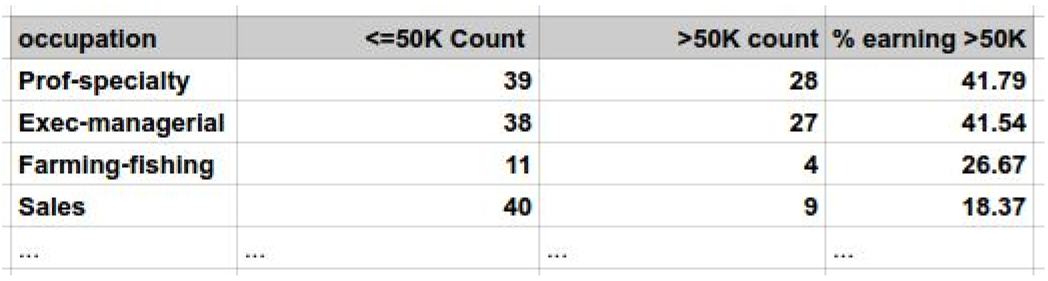
Another interesting pattern can be observed by looking at the **education**, **class** and **age** features together. If we look at all the people earning more than 50K with high school education or lesser, more than 75% of them are at least 40 years old. If we look at the same for people with more than high school education, more than 75% of them have an age between 30 and 50. The age histograms can be seen in ***Figure 4***.

Figure 4



Looking at the **occupation** and **class** features together, we can list down the occupations that have more percentage of people earning more than 50K. From ***Figure 5*** we can say that more than 40% of people working in Exec-managerial, Prof-speciality earn more than 50K.

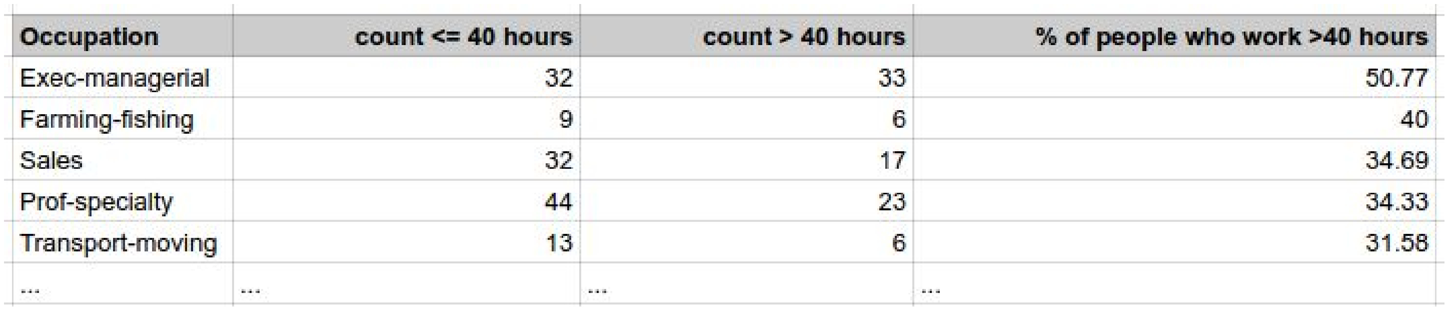
Figure 5



occupation	<=50K Count	>50K count	% earning >50K
Prof-specialty	39	28	41.79
Exec-managerial	38	27	41.54
Farming-fishing	11	4	26.67
Sales	40	9	18.37
...	...	...	...

A similar analysis can be done by looking at the **occupation** and **hours_per_week** to identify which jobs are more hectic compared to others. From ***Figure 6*** we can see that more than 50% of people in the Exec-Managerial occupation work more than 40 hours a week. It is interesting to note that the occupations with more people earning above 50K are same as the occupations with more people working beyond 40 hours a week.

Figure 6

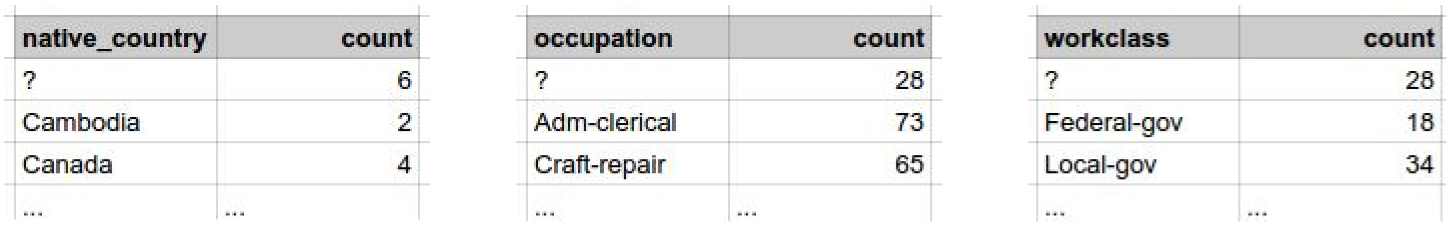


Occupation	count <= 40 hours	count > 40 hours	% of people who work >40 hours
Exec-managerial	32	33	50.77
Farming-fishing	9	6	40
Sales	32	17	34.69
Prof-specialty	44	23	34.33
Transport-moving	13	6	31.58
...	...	...	...

Missing data

Missing data for categorical attributes can be found out by looking at the histogram counts for each category. By applying this technique to the income dataset, we can determine the features that have missing data along with the number of records. We can see that the **native_country, workclass** and **occupation** features have missing values.

Figure 7



native_country	count
?	6
Cambodia	2
Canada	4
...	...

occupation	count
?	28
Adm-clerical	73
Craft-repair	65
...	...

workclass	count
?	28
Federal-gov	18
Local-gov	34
...	...

For ratio attributes like age and capital_gain, presence of missing data was determined by looking for non numerical values like ?, null, na, N/A. None of the ratio attributes in the income data set had any missing values.

Design and Implementation

Python programming language has been used for implementation. Numpy python library is used for performing vector level operations.

Iris dataset

The overall flow of the program for finding the k similar records in the iris dataset is as follows:

- Iris.csv is read and stored as a 2D numpy array
- All the four numerical columns from that dataset are normalized so that their values range between 0 and 1. **Min-Max normalization** was used to normalize the data. The minimum value in each column is mapped to zero, the maximum value is mapped to 1. The remaining data is scaled accordingly. Numpy.**vectorize** method was used to avoid any explicit for loops to iterate through the column values.

$$new\_value = \frac{(value - current\_min) * (new\_max - new\_min)}{(current\_max - current\_min)} + new\_min$$

- Since the Iris dataset doesn't contain any missing values, no special handling was done
- No transformation of the attributes was done as none of the attributes had a skewed distribution.
- For every record in the Iris dataset, the euclidean distance from every other record was computed. The program has an implementation for **Minkowski distance**. Euclidean distance is was computed by calculating Minkowski distance with **r=2**.

$$Minkowski\ distance = \left(\sum_{i=1}^{n} \left|x_i - y_i\right|^r\right)^{1/r}$$

- Euclidean distances are then converted to similarity values using the following equation:

$$similarity = \frac{1}{(1 + euclidean\ distance)}$$

- All the similarity scores are stored in a NxN matrix where N represents the number of records in the dataset. The tuple (x,y) at the row i and column j represents the (j+1)th similar record for record i, where x is the record number of the (j+1)th similar record and y is the similarity score.

- All the rows in the NxN matrix are then sorted such that the first k columns in each row i represents the first k similar records for the ith record in the dataset.
- The first k columns of the NxN matrix is then printed.

Income dataset

The overall flow of the program for finding the k similar records in the iris dataset is as follows:

- Income.csv is read and stored in a 2D array. We skip the **ID**, **education** and **class** column. Since **ID** has a unique nominal value for each record, it will not contribute in any way while computing similarity. **Education** column is skipped because the **education_cat** column in the dataset captures the same information as well.
- The values of features (workclass, marital_status, occupation, relationship, race, gender, native_country) with **nominal data type are mapped to numerical values**. We find all the unique values of the nominal feature, map it to an integer starting from 0 with unit increments. For e.g, the unique values of the gender column Male and Female will be mapped to 0 and 1 respectively.
- The values of the ordinal column of **education_cat** are also adjusted to make sure it ranges from 0-15 instead of 1-16. This makes it easier to compute ordinal similarity.
- **Missing values** in the any of the ordinal or nominal columns are replaced with a **'nan'** string. This ensures that the array can be type converted to **float** without any errors. Numpy replaces these **'nan'** strings with a special value **np.nan**. We can use the **np.isnan** to detect these **nan** values.
- The values of the **captial_gain** column are then **log transformed**. In our exploratory data analysis we saw that the values of this column are left skewed and normalization without log transformation will lead to very less values being assigned to the values in the head of the distribution. Before performing log transformation, we add one to all the existing values to change all zeros to one.
- All the columns in the dataset now contain only numerical values. The entire 2D array is then converted to float using the **astype** method provided by numpy. As mentioned above all the cells with missing values now have **nan** special value provided by numpy
- All the columns holding ratio attributes (age, fnlwgt, captial_gain, captial_loss, hours_per_week) are now normalized to 0-1 range using **min-max normalization**.
- Similar to the Iris dataset, we maintain a NxN matrix with the similarity scores, where N is the number of records in the dataset. We perform the following similarity computations for every pair of records

- We take the values of the ratio attributes of both the records. We look at the corresponding values for each feature in both the records, if either record's value is **nan**, we skip that feature in both the records. This is implemented in the **exclude_missing_data_columns** method. The output of the method is the two new vectors with only features that are not missing for both the records. We make a note of the number of features in the new vectors for later use. Cosine similarity is then computed for the new vectors. After computing similarity, the score is then multiplied by the number of ratio features that hold valid data for two records.

$$cosine\_similarity = \frac{d_1 \bullet d_2}{|d_1| \times |d_2|}$$

d_1 and d_2 - ratio attribute vectors

$d_1 \bullet d_2$ - dot product of vectors

$|d|$ - magnitude of vector d

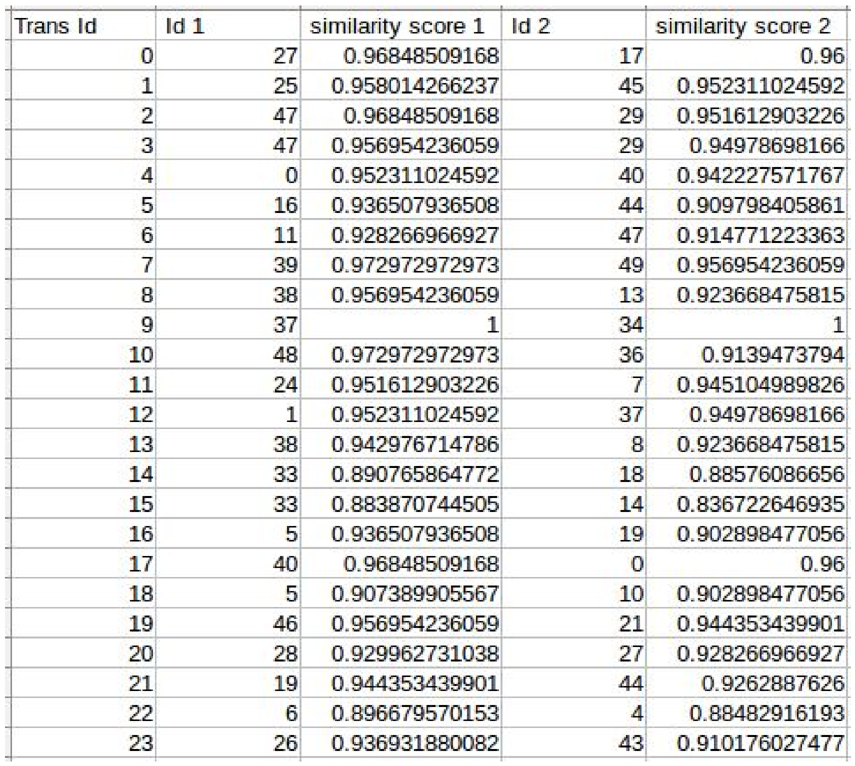
- We take the values of the nominal attributes of both the records and perform a simple equality check between the corresponding attribute values and map it to 1 if they are equal and 0 otherwise. We take the sum of the nominal similarity scores of all the attributes and make a note of it further use. **Missing values** are handled in manner similar to that of ratio attributes, we skip all the features where either one of the records have a missing value. We make a note of the number of features that have valid values for both the records
- For the **education_cat** ordinal attribute, we compute the ordinal similarity between two features using the following formula, where the number of values is 16

$$ordinal\_similarity = \frac{record\_1\ value - record\_2\ value}{(number\ of\ values - 1)}$$

- We then compute the overall similarity for the entire record by adding the cosine similarity value, total nominal similarity value and the ordinal similarity value and divide it by the number of features that did not have any missing values for the pair of records. This ensures that the overall similarity value is between 0 and 1. We make note of the similarity between the i^{th} record and the j^{th} record at the (i,j) cell in the NxN similarity score matrix.
- After computing similarity scores for all pairs of records, we sort all the rows in descending order with respect to similarity score and print the first k columns of the matrix.

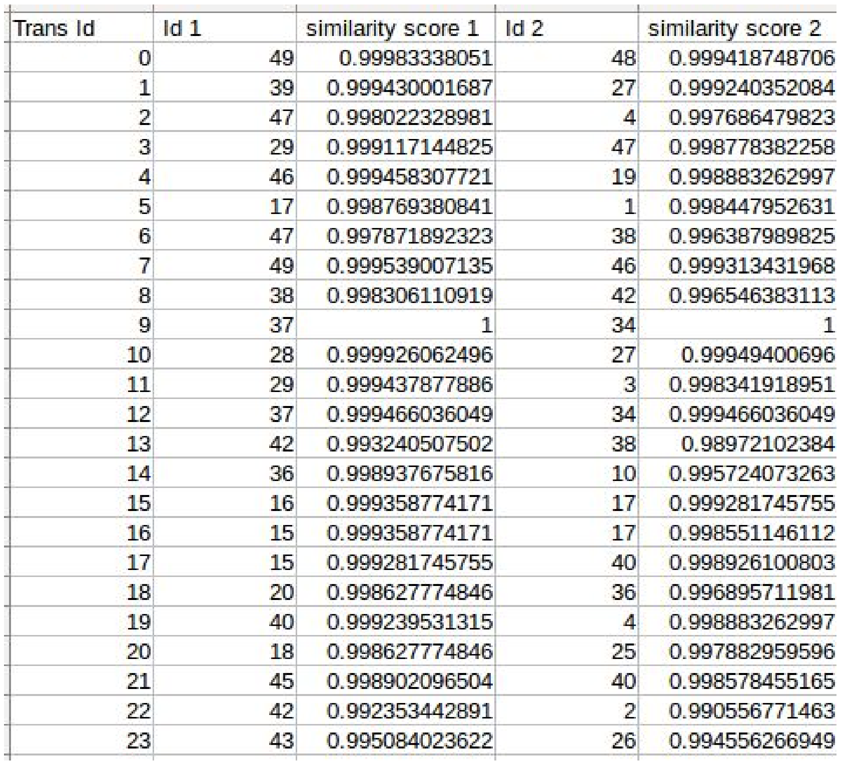
Iris dataset output

The first 23 rows are shown in the output, Euclidean distance measure was used to determine similarity for this output:



Trans Id	Id 1	similarity score 1	Id 2	similarity score 2
0	27	0.96848509168	17	0.96
1	25	0.958014266237	45	0.952311024592
2	47	0.96848509168	29	0.951612903226
3	47	0.956954236059	29	0.94978698166
4	0	0.952311024592	40	0.942227571767
5	16	0.936507936508	44	0.909798405861
6	11	0.928266966927	47	0.914771223363
7	39	0.972972972973	49	0.956954236059
8	38	0.956954236059	13	0.923668475815
9	37	1	34	1
10	48	0.972972972973	36	0.9139473794
11	24	0.951612903226	7	0.945104989826
12	1	0.952311024592	37	0.94978698166
13	38	0.942976714786	8	0.923668475815
14	33	0.890765864772	18	0.88576086656
15	33	0.883870744505	14	0.836722646935
16	5	0.936507936508	19	0.902898477056
17	40	0.96848509168	0	0.96
18	5	0.907389905567	10	0.902898477056
19	46	0.956954236059	21	0.944353439901
20	28	0.929962731038	27	0.928266966927
21	19	0.944353439901	44	0.9262887626
22	6	0.896679570153	4	0.88482916193
23	26	0.936931880082	43	0.910176027477

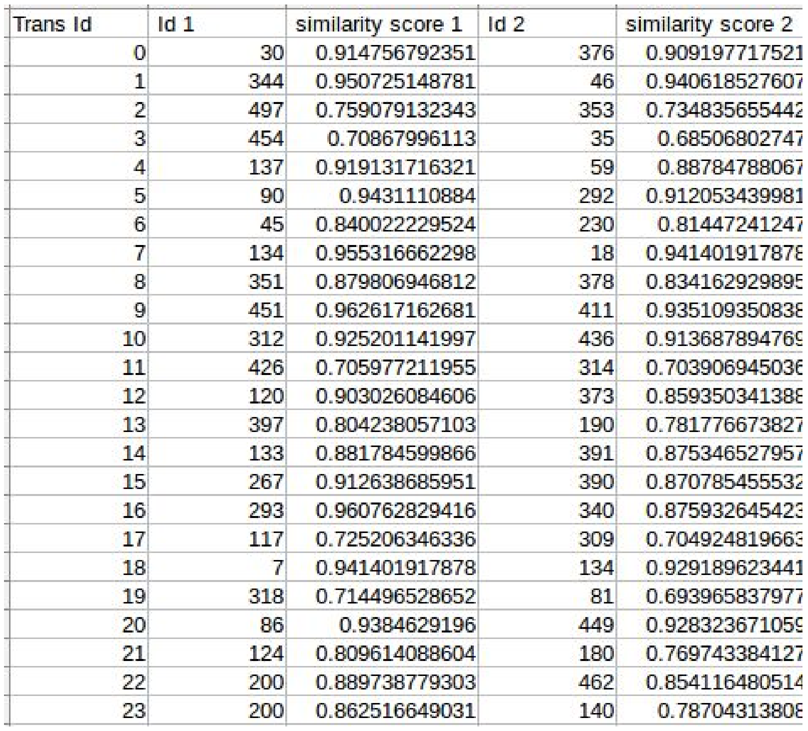
Cosine similarity was used to get the k=2 similar records in the following output:



Trans Id	Id 1	similarity score 1	Id 2	similarity score 2
0	49	0.99983338051	48	0.999418748706
1	39	0.999430001687	27	0.999240352084
2	47	0.998022328981	4	0.997686479823
3	29	0.999117144825	47	0.998778382258
4	46	0.999458307721	19	0.998883262997
5	17	0.998769380841	1	0.998447952631
6	47	0.997871892323	38	0.996387989825
7	49	0.999539007135	46	0.999313431968
8	38	0.998306110919	42	0.996546383113
9	37	1	34	1
10	28	0.999926062496	27	0.99949400696
11	29	0.999437877886	3	0.998341918951
12	37	0.999466036049	34	0.999466036049
13	42	0.993240507502	38	0.98972102384
14	36	0.998937675816	10	0.995724073263
15	16	0.999358774171	17	0.999281745755
16	15	0.999358774171	17	0.998551146112
17	15	0.999281745755	40	0.998926100803
18	20	0.998627774846	36	0.996895711981
19	40	0.999239531315	4	0.998883262997
20	18	0.998627774846	25	0.997882959596
21	45	0.998902096504	40	0.998578455165
22	42	0.992353442891	2	0.990556771463
23	43	0.995084023622	26	0.994556266949

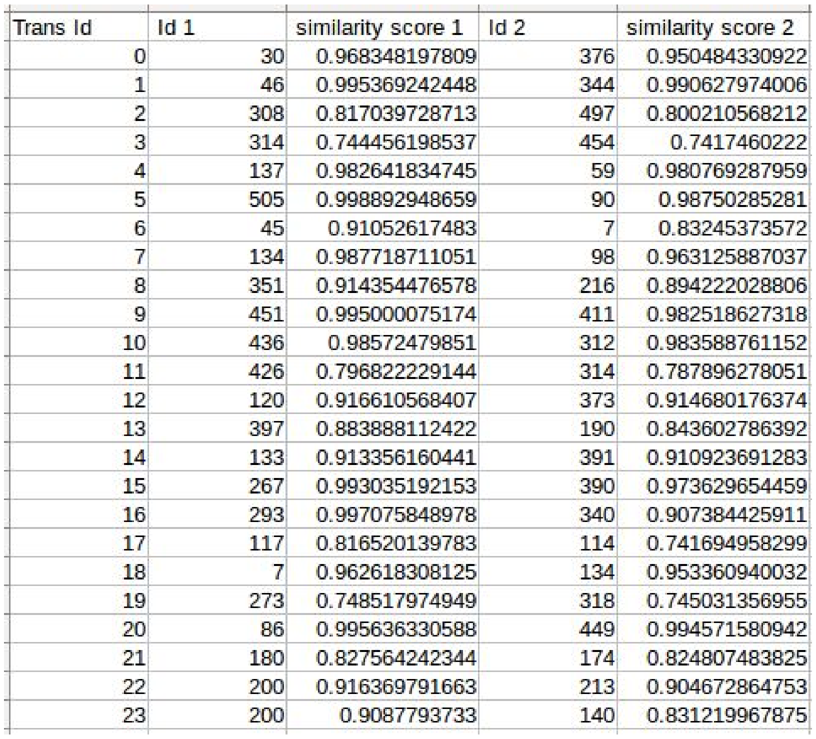
Income dataset output

The following is the output of the program for the Income dataset with k=2 and euclidean distance being used to measure similarity



Trans Id	Id 1	similarity score 1	Id 2	similarity score 2
0	30	0.914756792351	376	0.909197717521
1	344	0.950725148781	46	0.940618527607
2	497	0.759079132343	353	0.734835655442
3	454	0.70867996113	35	0.68506802747
4	137	0.919131716321	59	0.88784788067
5	90	0.9431110884	292	0.912053439981
6	45	0.840022229524	230	0.81447241247
7	134	0.955316662298	18	0.941401917878
8	351	0.879806946812	378	0.834162929895
9	451	0.962617162681	411	0.935109350838
10	312	0.925201141997	436	0.913687894769
11	426	0.705977211955	314	0.703906945036
12	120	0.903026084606	373	0.859350341388
13	397	0.804238057103	190	0.781776673827
14	133	0.881784599866	391	0.875346527957
15	267	0.912638685951	390	0.870785455532
16	293	0.960762829416	340	0.875932645423
17	117	0.725206346336	309	0.704924819663
18	7	0.941401917878	134	0.929189623441
19	318	0.714496528652	81	0.693965837977
20	86	0.9384629196	449	0.928323671059
21	124	0.809614088604	180	0.769743384127
22	200	0.889738779303	462	0.854116480514
23	200	0.862516649031	140	0.78704313808

The following was obtained by using cosine similarity measure between ratio attributes



Trans Id	Id 1	similarity score 1	Id 2	similarity score 2
0	30	0.968348197809	376	0.950484330922
1	46	0.995369242448	344	0.990627974006
2	308	0.817039728713	497	0.800210568212
3	314	0.744456198537	454	0.7417460222
4	137	0.982641834745	59	0.980769287959
5	505	0.998892948659	90	0.98750285281
6	45	0.91052617483	7	0.83245373572
7	134	0.987718711051	98	0.963125887037
8	351	0.914354476578	216	0.894222028806
9	451	0.995000075174	411	0.982518627318
10	436	0.98572479851	312	0.983588761152
11	426	0.796822229144	314	0.787896278051
12	120	0.916610568407	373	0.914680176374
13	397	0.883888112422	190	0.843602786392
14	133	0.913356160441	391	0.910923691283
15	267	0.993035192153	390	0.973629654459
16	293	0.997075848978	340	0.907384425911
17	117	0.816520139783	114	0.741694958299
18	7	0.962618308125	134	0.953360940032
19	273	0.748517974949	318	0.745031356955
20	86	0.995636330588	449	0.994571580942
21	180	0.827564242344	174	0.824807483825
22	200	0.916369791663	213	0.904672864753
23	200	0.9087793733	140	0.831219967875

Analysis of output

From the output screenshots above, we can see that for most of the cases the similarity scores of the first and second closest neighbors are very close to each other. We can run the program for different values of k and look at how the scores vary as k-value increases.

Figure 8

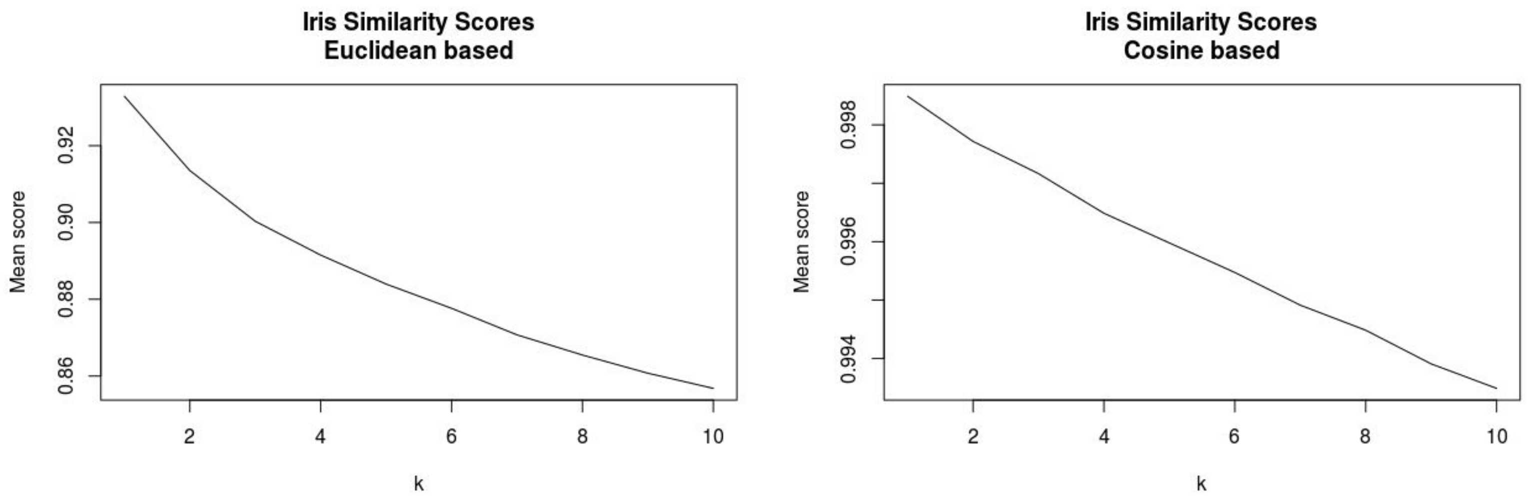
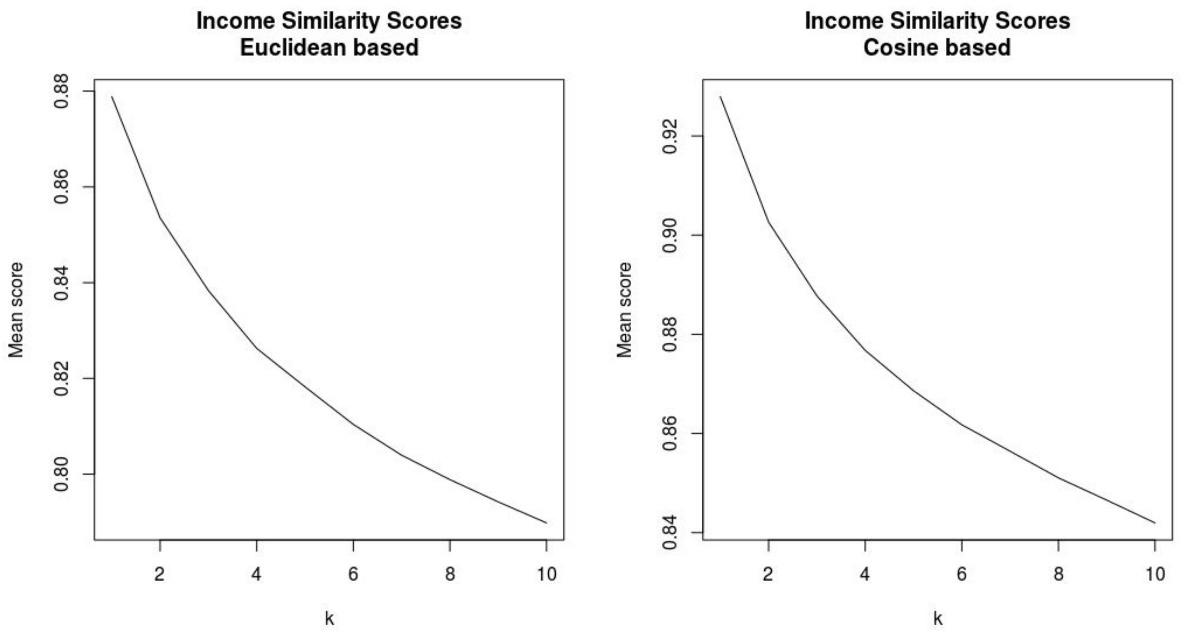


Figure 8 shows trend for similarity scores in the Iris dataset, the program was run with the **k=10**. Along the y-axis, we have the average of the similarity score of all the k-th similar records. We can see that as the value of k increases the similarity score also drops gradually. We can also see that the similarity scores based on **Euclidean distance** seem to drop at a faster rate when compared to the **Cosine Similarity** scores, with the cosine similarity scores still being very close 0.9 for k=10.

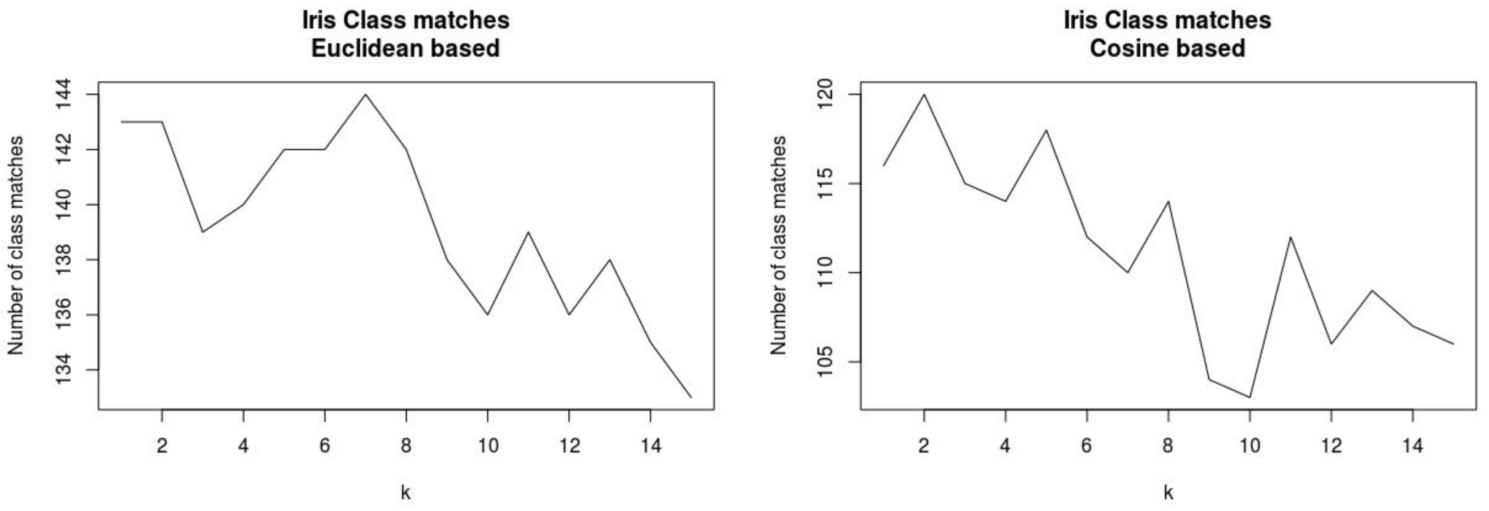
Figure 9



We can look at similar metrics for the Income data as well. In ***Figure 9***, we can see the score trend for income dataset with increasing values of k. The similarity scores seem to drop at a higher rate when compared to the Iris dataset. It is interesting to note that **Cosine Similarity** scores also decreasing at a faster rate when compared to the **Cosine Similarity** scores of the Iris dataset. Iris dataset only contains 4 features whereas the Income dataset has around 14 features which could be the reason for the faster rate of decrease of similarity.

We can also compare the class attribute of the record and its kth nearest neighbors and see if it matches. We can do this for different values of k and look how the number of matches varies along with k.

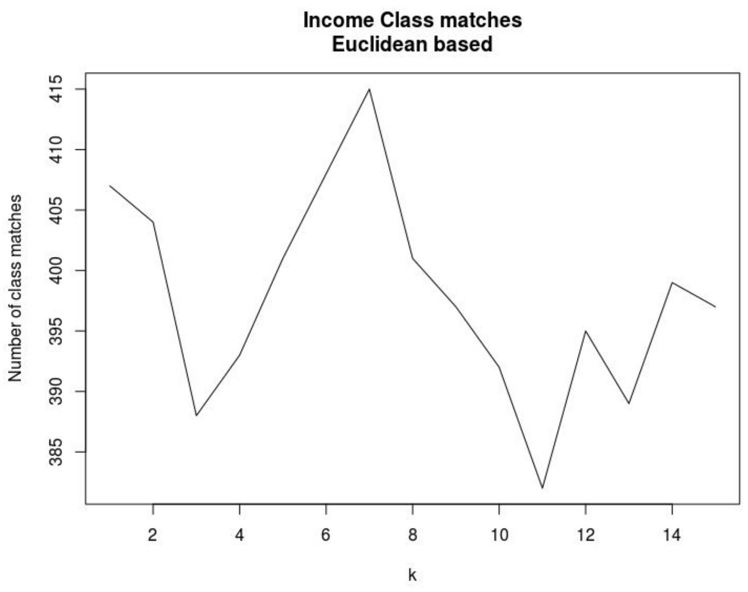
Figure 10



In ***Figure 10***, we plot the number of records for which the class attribute matches with its kth nearest neighbor. We can see trend similar to the one we saw in the ***Figure 8***. There is a general decrease in the number of records for which the class attributes match. However, it is interesting to note that it is not a monotonically decreasing curve. In the “Euclidean based” graph, there is a decrease in the number of matches for k=3, after which the number of matches increases until k=7 and then drops suddenly at k=10, goes up and down again after that. This could mean that there are few records that are in a “fuzzy” neighborhood, where there are comparable number of matching and non-matching classes. Increasing the value of k includes a matching class record or a non matching with equal probability.

We can see a similar trend in both Euclidean based and Cosine based distance measures. In both cases, there seems to be a sudden drop in the number of matches at 10th nearest neighbor. Cosine similarity graph also has the same saw-tooth shaped graph after k=10, which further bolsters the speculation that few records could be in a “fuzzy” neighborhood.

Figure 11



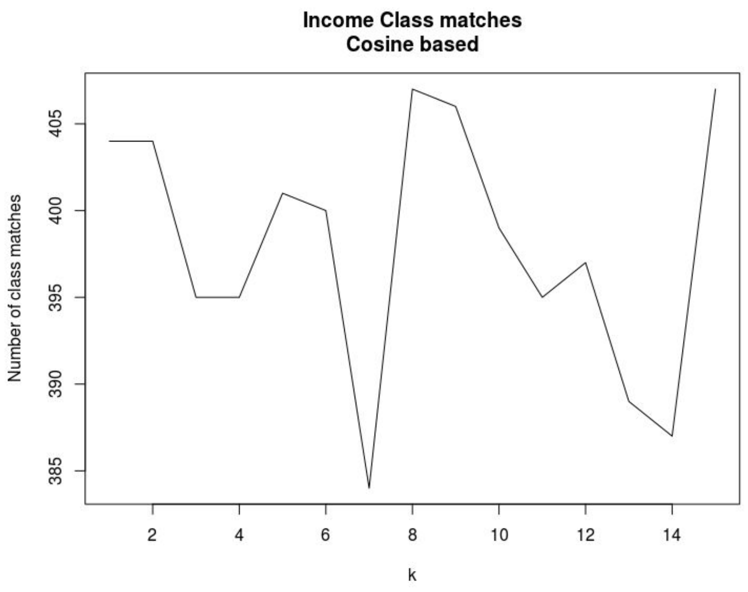


Figure 11 shows a similar class match trend plot for the Income dataset. However, there doesn't seem to be any clear trend on the number of matches. There is an initial decrease as the value of k increases to 3, but after that the Euclidean based graph has huge surge in the number of class matches. At k=7, the number of class matches is even higher than k=1, we can see a similar peak in the Cosine Similarity graph as well at k=8 and also at k=15. We can also notice a dip in the number of matches at k=11 in both the graphs. However, there is drastic decrease in the euclidean based graph when compared to the Cosine Similarity one. This could mean that there are many records which could farther apart in euclidean distance but they could be pointing along the same direction which leads to the improved Cosine similarity score.