PERFORMANCE ANALYSIS AND WINNING PREDICTION OF ATHLETES IN OLYMPIC GAMES USING MACHINE LEARNING ALGORITHMS.

FOR THE FULFILLMENT OF J-COMPONENT FOR DATA MINING AND BUSINESS INTELLIGENCE

(ITA- 5007)

PROJECT REPORT

SUBMITTED BY :

UMANG JAIN- 19MCA0164

RAHUL SHARMA- 19MCA0174

AKSHIT- 19MCA0020

ABSTRACT

Olympic games has been a major center of attraction of people around the globe for a long time now. It is a platform for athletes around the globe to come and showcase their talents on an international platform.Olympic games includes a wide range of activities and games with athletes from various parts of the world competing to bring fame to their country.

In this project we are trying to build a machine learning model which would attempt to perform a prediction on the chance of an athlete to win a medal in any activity or game. Further we will try perform some exploratory data analysis on the performance of various athletes in different games of Olympics.

LITERATURE SURVEY

In *"Models and Data Mining Algorithms for Solving Classification Problems"* [1] the authors focussed on the application of the method of data mining - support vector machine (SVM) to solve the practical problem of evaluating the efficiency of oil wells. This nonlinear method showed better analysis results than the linear regression (LR) method, which is also a machine learning method. The paper presented and analysed the principles of solving the classification problem using logistic regression methods and support vector machines. The paper presented a theoretical analysis of the support vector machine and logistic regression. It is shown, that the nonlinear SVM algorithm works better than the linear LR algorithm in the analysis of the oil well system and forecasting their efficiency. The paper *"A Review on Different Data Mining Algorithms and Selection Methods"* [2] overviews diverse parts of data mining research. The paper also talks about the algorithm selection issue in data mining. There are seven stages in data mining. They are data cleaning, data coordination, data selection, data change, data mining, knowledge introduction, and example development. Another paper titled *"Research on Intelligent Tutoring System Based on Data-Mining Algorithms"* [3] This paper takes the composition of the intelligent teaching system and the realization process of the system as study core, to introduce the process of the design and realization of each component of the intelligent tutoring system. It adopts data mining association rules and decision tree mining algorithm to enhance the intelligence and personalization of the intelligent tutoring system. In the course of the design, the thought of evaluating the students' learning effect by changing the traditional expert model is used, according to the constructivist teaching theory, and the expert model is adjusted based on the students' learning effect. The experimental results show that the application of data mining technology to the decision-making of teaching management in colleges enable the decision makers to be more pertinent in decision making, and it has certain practical guiding significance. In the paper *"Performance Evaluation of Mahout Clustering Algorithms Using a Twitter Streaming Dataset"* [4] the authors analysed the performance of some clustering algorithms of Apache Mahout using a Twitter streaming dataset under a Hadoop MapReduce cluster infrastructure according to various evaluation criteria. The authors mentioned about several Cloud-based implementations of machine learning (ML) and data mining (DM) algorithms being emerging after the Big Data. Such implementations aim to overcome limitations of traditional ML and DM frameworks to handle Big Data. Mahout is one such Cloud-based implementation of ML and DM algorithms to efficiently deal with Big Data. Among interesting algorithms there are the clustering algorithms whose performance is affected by the number of entries in the data set. We have presented some performance evaluation results for Mahout cluster algorithms using Twitter Stream data set. We observed that significant reduction in processing time can be achieved for clustering algorithms executed on Hadoop MapReduce. In another work titled *"Enhanced mining association rule algorithm with reduced time & space complexity"* [5] the authors have proposed a technique to improve the performance of existing mining association rule algorithm which significantly reduces the time and space complexity of independent of datasets. There are many data mining algorithms for finding association rules our contribution can be used in almost all of the algorithms independent of its variety. In this paper we are concentrating more on Apriori algorithm which is a type of candidate generation algorithm also a fundamental block of all the mining algorithms, rectifying its major limitation of consuming ample amount of time in generating the candidates. In *"Improved algorithm based on Sequential pattern mining of big data set"* [6], the paper improved Prefix Span algorithm and proposes BLSPM algorithm. This algorithm can greatly reduce the numbers of construction projection database, thus improving the efficiency of sequential pattern mining.

PROBLEM DESCRIPTION

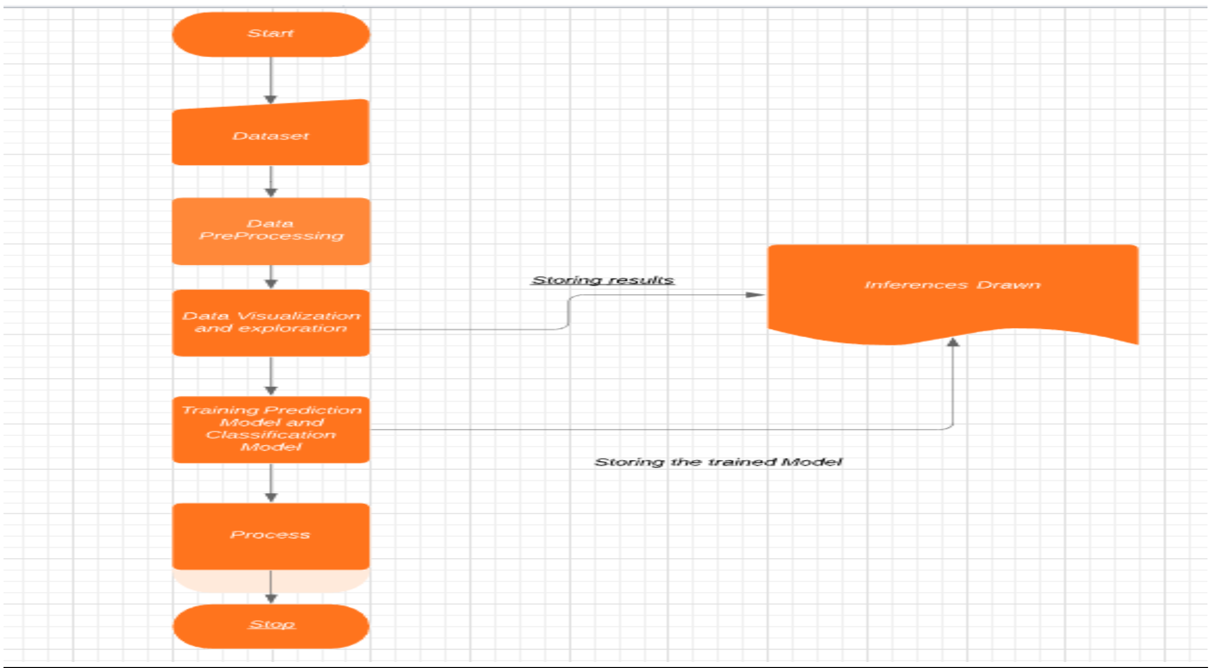
Olympic games has been a major center of attraction of people around the globe for a long time now. It is a platform for athletes around the globe to come and showcase their talents on an international platform.Olympic games includes a wide range of activities and games with athletes from various parts of the world competing to bring fame to their country.

In this project we are trying to build a machine learning model which would attempt to perform a prediction on the chance of an athlete to win a medal in any activity or game. Further we will try perform some exploratory data analysis on the performance of various athletes in different games of Olympics.

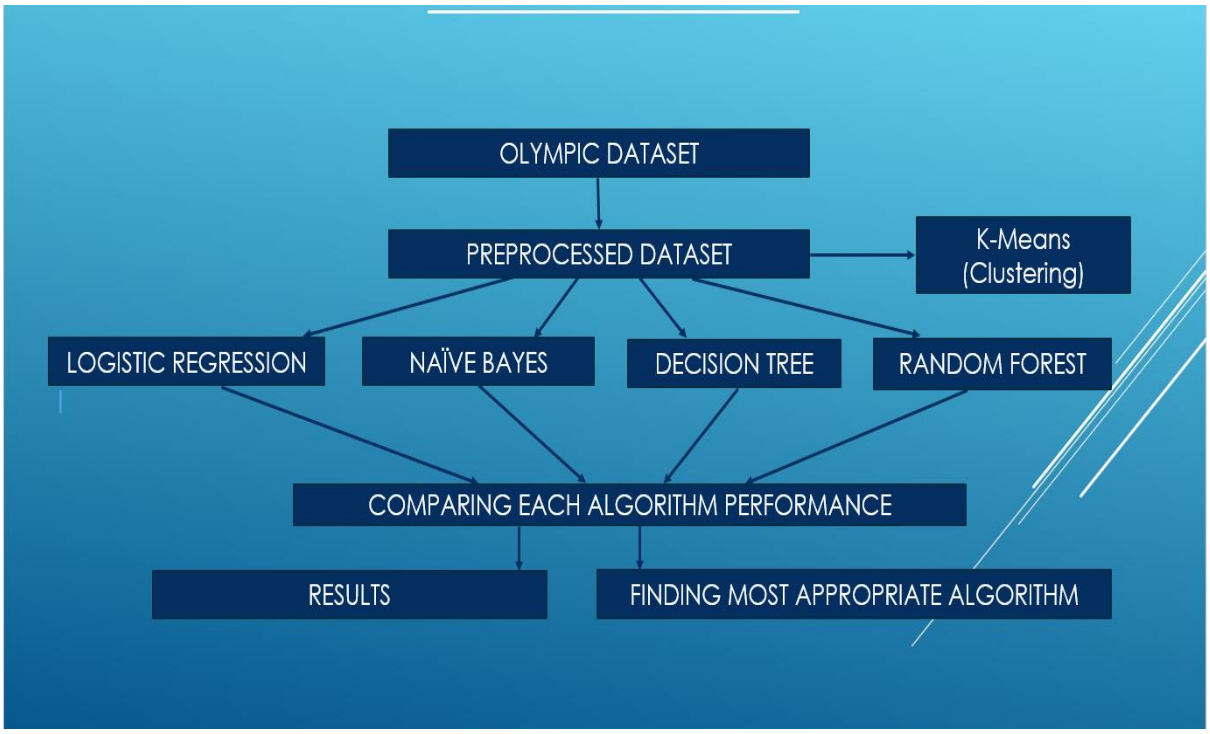
There has been a need to predict the winning probability of the athletes taking part in any event of the Olympic games based on the characteristics of the athlete-age, height, weight, sport, country, etc.

Here, we are trying to fulfil this issue by building a machine learning model using appropriate machine learning algorithm.

ARCHITECTURE DIAGRAM

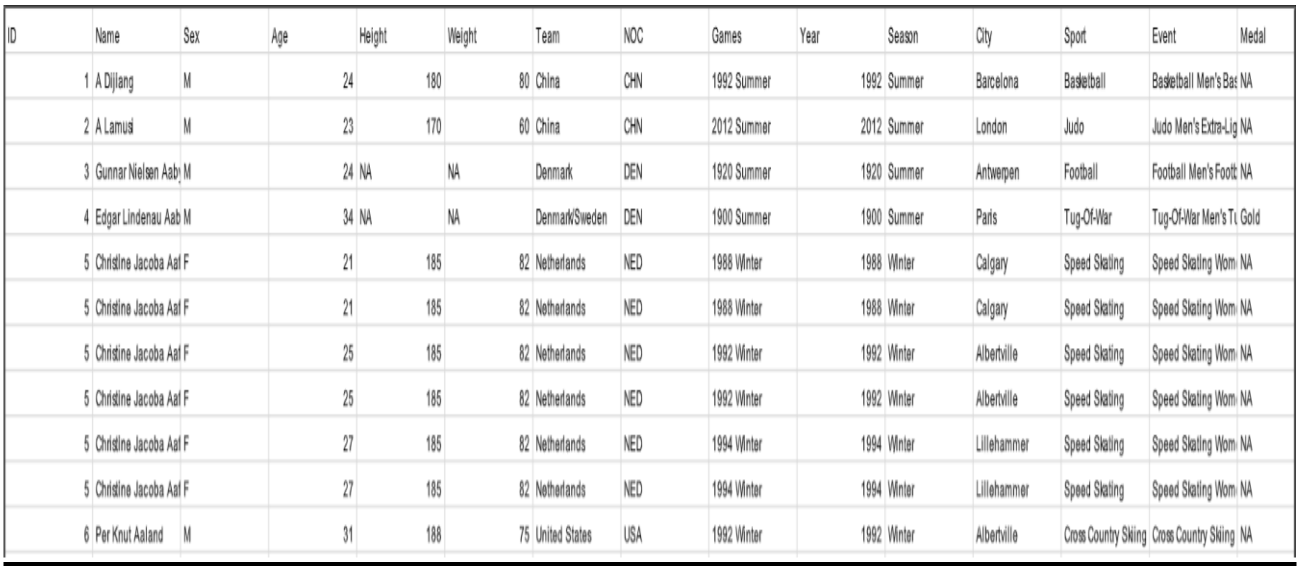

Start
Dataset
Data PreProcessing
Data Visualization and exploration
Training Prediction Model and Classification Model
Process
Stop
Storing results
Inferences Drawn
Storing the trained Model

DETAILED DESIGN



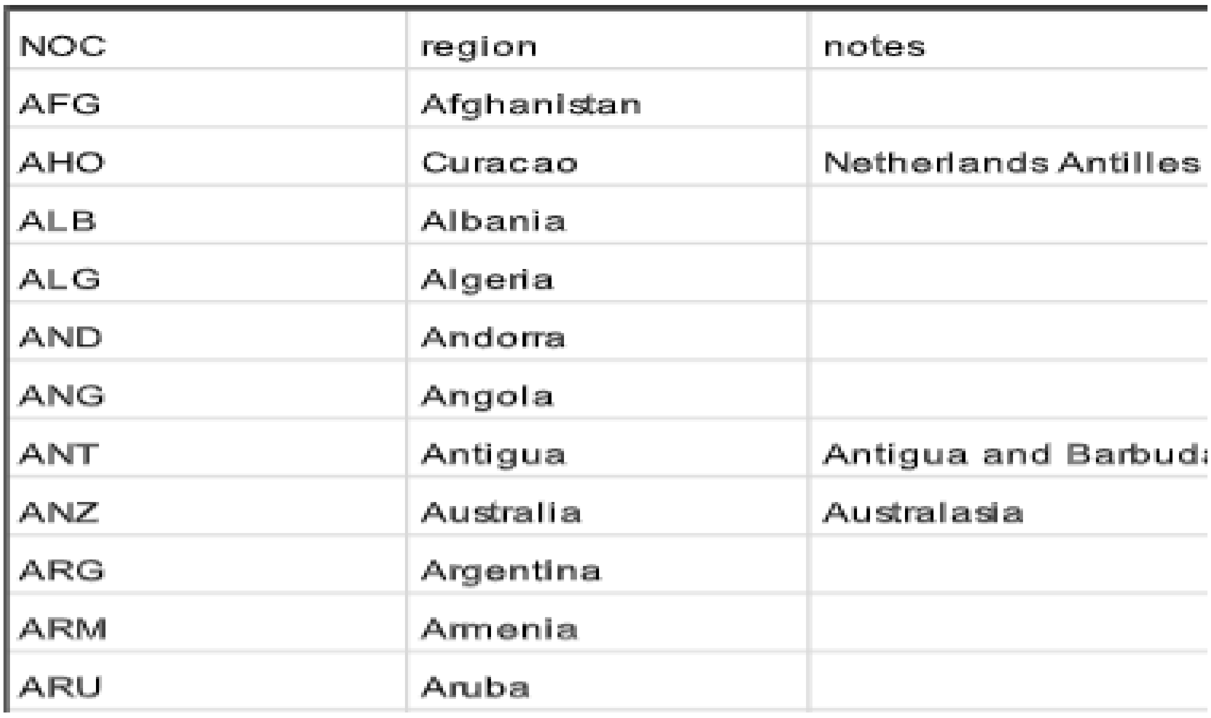
DATASETS AVAILABLE

DATASET-1 (athlete_events.csv)



ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
1	A Dijiang	M	24	180	80	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Bas	NA
2	A Lamusi	M	23	170	60	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lig	NA
3	Gunnar Nielsen Aaby	M	24	NA	NA	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Footb	NA
4	Edgar Lindenau Aab	M	34	NA	NA	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tu	Gold
5	Christine Jacoba Aaf	F	21	185	82	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Wom	NA
5	Christine Jacoba Aaf	F	21	185	82	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Wom	NA
5	Christine Jacoba Aaf	F	25	185	82	Netherlands	NED	1992 Winter	1992	Winter	Albertville	Speed Skating	Speed Skating Wom	NA
5	Christine Jacoba Aaf	F	25	185	82	Netherlands	NED	1992 Winter	1992	Winter	Albertville	Speed Skating	Speed Skating Wom	NA
5	Christine Jacoba Aaf	F	27	185	82	Netherlands	NED	1994 Winter	1994	Winter	Lillehammer	Speed Skating	Speed Skating Wom	NA
5	Christine Jacoba Aaf	F	27	185	82	Netherlands	NED	1994 Winter	1994	Winter	Lillehammer	Speed Skating	Speed Skating Wom	NA
6	Per Knut Aaland	M	31	188	75	United States	USA	1992 Winter	1992	Winter	Albertville	Cross Country Skiing	Cross Country Skiing	NA

<u>DATASET-2</u> (NOC_regions.csv)



NOC	region	notes
AFG	Afghanistan	
AHO	Curacao	Netherlands Antilles
ALB	Albania	
ALG	Algeria	
AND	Andorra	
ANG	Angola	
ANT	Antigua	Antigua and Barbud
ANZ	Australia	Australasia
ARG	Argentina	
ARM	Armenia	
ARU	Aruba	

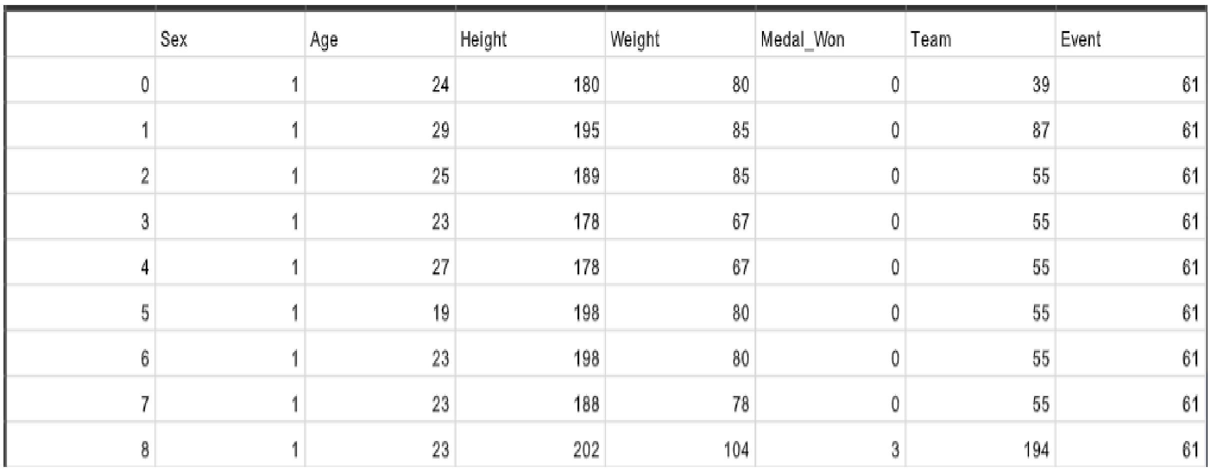
<u>DATA PRE-PROCESSING</u>

Checking for Null values in the dataset, we find there are a lot of Null values or NA in the Age, Height, weight and Medal Columns.

- For Medals column, NA means athlete did not win the medal. So converting NA to DNW (did not win). Further assigning 1 to all numerical cells and 0 to all DNW cells.
- For Age, Height and Weight Columns, substituting Null values with the average of these fields of their particular Events.
- Keeping values of only summer Olympics after 1960 under consideration and dropping other records.
- Dropping unnecessary columns like Name, Games, season, City, sport.

PRE-PROCESSED DATASET

After performing the required pre-processing operations on the dataset, we get the final pre-processed dataset to work on as under:



	Sex	Age	Height	Weight	Medal_Won	Team	Event
0	1	24	180	80	0	39	61
1	1	29	195	85	0	87	61
2	1	25	189	85	0	55	61
3	1	23	178	67	0	55	61
4	1	27	178	67	0	55	61
5	1	19	198	80	0	55	61
6	1	23	198	80	0	55	61
7	1	23	188	78	0	55	61
8	1	23	202	104	3	194	61

METHODOLOGIES USED

In this project, we are trying to build an appropriate model for predicting the probability of an athlete winning a medal in Olympic games, provided certain information about the candidate. We are using the past record of Olympic games as the training data to train our model to give appropriate results. In our project, we have performed the following tasks:

- Pre-processed the data using appropriate methods.
- Applied clustering using K-Means Clustering algorithm on the data and plotted a scatter plot to better visualise out data.
- Performed classification and built a model to give the required results using four major machine learning algorithms which include- Logistic Regression Algorithm, Decision Tree Algorithm, Naïve Bayes Algorithm and Decision Tree Algorithm.

- Last, we have tried to compare the scores of these algorithms to conclude at a decision as to which algorithm is most efficient in respect to the scenario in hand. For measuring the efficiency of the algorithms, we have taken into account different metrics including Accuracy score, Confusion Matrix, etc.

CODING

1) PreProcessing.ipynb:

```
import pandas as pd
import numpy as np
from sklearn import preprocessing


olympics=pd.read_csv('/content/drive/My Drive/DataMiningProjecct/athlet
e_events.csv')


#Printing column wise missing values
print(olympics.isnull().sum())


#replace these missing values in medals by 'Did not win' or 'DNW' for t
he atheletes who did not win any medals
olympics['Medal'].fillna('DNW', inplace = True)


#reading NOC dataset
noc_country = pd.read_csv('/content/drive/My Drive/DataMiningProjecct/n
oc_regions.csv')
noc_country.drop('notes', axis = 1 , inplace = True)
noc_country.rename(columns = {'region':'Country'}, inplace = True)


#joining both datasets with NOC as primary key
olympics_merge = olympics.merge(noc_country,
                                left_on = 'NOC',
                                right_on = 'NOC',
                                how = 'left')
```

```
# Do we have NOCs that didnt have a matching country in the master?
print(olympics_merge.loc[olympics_merge['Country'].isnull(),['NOC', 'Te
am']].drop_duplicates())


# Replace missing Teams by the values 1. SGP - Singapore
                                    # 2. ROT - Refugee Olympic Athletes
                                    # 3. UNK - Unknown
                                    # 4. TUV - Tuvalu
#olympics_merge.loc[olympics_merge['Country'].isnull(), ['Country']] =
olympics_merge['Team']

olympics_merge['Country'] = np.where(olympics_merge['NOC']=='SGP', 'Sin
gapore', olympics_merge['Country'])
olympics_merge['Country'] = np.where(olympics_merge['NOC']=='ROT', 'Ref
ugee Olympic Athletes', olympics_merge['Country'])
olympics_merge['Country'] = np.where(olympics_merge['NOC']=='UNK', 'Unk
nown', olympics_merge['Country'])
olympics_merge['Country'] = np.where(olympics_merge['NOC']=='TUV', 'Tuv
alu', olympics_merge['Country'])


# Drop Team column and rename Country column to team column
olympics_merge.drop('Team', axis = 1, inplace = True)
olympics_merge.rename(columns = {'Country': 'Team'}, inplace = True)


#Checking again for mapping of NOC to team we find that each is mapped
to a single value.
print(olympics_merge.loc[olympics_merge['Team'].isnull(),['NOC', 'Team'
]].drop_duplicates())


#checking Null Values
print(olympics_merge.isnull().sum())


# Lets take data from 1961 onwards only and for summer olympics only
olympics_complete_subset = olympics_merge.loc[(olympics_merge['Year'] >
 1960) & (olympics_merge['Season'] == "Summer"), :]
print(olympics_complete_subset.head())


# We see the row indices are not contigent anymore due to removal of ro
ws. So,Reset row indices
olympics_complete_subset = olympics_complete_subset.reset_index()
print(olympics_complete_subset.head())


#checking Null Values
```

```
print(olympics_complete_subset.isnull().sum())
#Null values reduced considerably by using the method of discarding tup
les


print(olympics_complete_subset)


#Extracting unique events in a new list
listunique=olympics_complete_subset.Event.unique()
print(listunique)
print(len(listunique))


tempagemean=[]
tempweightmean=[]
tempheightmean=[]

#looping to the event list and creating new lists for means of age,weig
ht and height for unique events.
for i in listunique:
    temporary=olympics_complete_subset.loc[(olympics_complete_subset['E
vent'] == i) & (olympics_complete_subset['Age'] !='NaN' ), :]
    tempagemean.append(temporary.Age.mean(axis = 0, skipna = True))
    tempweightmean.append(temporary.Weight.mean(axis=0,skipna = True))
    tempheightmean.append(temporary.Height.mean(axis = 0, skipna = True
))


#looping for each event(since there are 372 events)
for i in range(0,372):
    #extracting rows with a particular event into another dataframe
    df1 = olympics_complete_subset[olympics_complete_subset['Event'].st
r.contains(listunique[i]) ]
    #filling null values for age in the new dataframe to the mean age f
or that event
    df1.Age.fillna(tempagemean[i],inplace=True)
    # filling null values for weight in the new dataframe to the mean w
eight for that event
    df1.Weight.fillna(tempweightmean[i], inplace=True)
    # filling null values for height in the new dataframe to the mean h
eight for that event
    df1.Height.fillna(tempheightmean[i], inplace=True)
    #Dropping columns for that particular event in the original datafra
me
    olympics_complete_subset.drop(olympics_complete_subset[olympics_com
plete_subset['Event'].str.contains(listunique[i])].index, inplace = Tru
e)
```

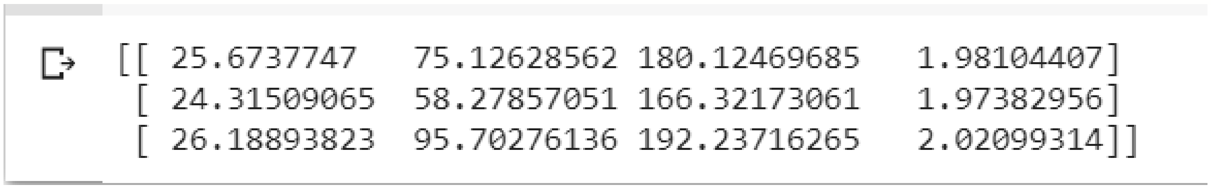
```
    #concatinating the altered rows in new dataframe to the old datafra
me
    olympics_complete_subset = pd.concat([olympics_complete_subset, df1
], axis=0)




#Print statements to check for null values
print(olympics_complete_subset.head())
print(olympics_complete_subset.isnull().sum())
print(olympics_complete_subset)


#giving numeric value to gender column 1 for male and 0 for female
olympics_complete_subset['Sex'] = np.where(olympics_complete_subset['Se
x']=='M', 1 , 0)


#Creating a column that captures whether or not a medal was won! It wou
ld be 1 if Medal column says Gold, Silver or Bronze and 0 otherwise.
#Converting Medal column to numeric form
olympics_complete_subset['Medal_Won'] = np.where(olympics_complete_subs
et.loc[:,'Medal'] == 'DNW', 0 , olympics_complete_subset['Medal'])
olympics_complete_subset['Medal_Won'] = np.where(olympics_complete_subs
et.loc[:,'Medal_Won'] == 'Bronze', 1 , olympics_complete_subset['Medal_
Won'])
olympics_complete_subset['Medal_Won'] = np.where(olympics_complete_subs
et.loc[:,'Medal_Won'] == 'Silver', 2 , olympics_complete_subset['Medal_
Won'])
olympics_complete_subset['Medal_Won'] = np.where(olympics_complete_subs
et.loc[:,'Medal_Won'] == 'Gold', 3 , olympics_complete_subset['Medal_Wo
n'])
print(olympics_complete_subset['Medal_Won'].unique())




#Dropping unnecessary columns from dataFrame
olympics_complete_subset.drop(['index'] , axis=1, inplace=True)
olympics_complete_subset.drop(['Name'] , axis=1, inplace=True)
olympics_complete_subset.drop(['NOC'] , axis=1, inplace=True)
olympics_complete_subset.drop(['Season'] , axis=1, inplace=True)
olympics_complete_subset.drop(['Games'] , axis=1, inplace=True)
olympics_complete_subset.drop(['City'] , axis=1, inplace=True)
olympics_complete_subset.drop(['Year'] , axis=1, inplace=True)
olympics_complete_subset.drop(['Sport'] , axis=1, inplace=True)
olympics_complete_subset.drop(['Medal'] , axis=1, inplace=True)
olympics_complete_subset.drop(['ID'] , axis=1, inplace=True)
```

```
# We see the row indices are not contigent anymore due to removal of ro
ws. So,Reset row indices
olympics_complete_subset = olympics_complete_subset.reset_index()


#Creating object of preprocessing LabelEncoder
le = preprocessing.LabelEncoder()
#Creating a new column of encoded Team and Encoded Event in numerical f
orm in original dataset
olympics_complete_subset['Team_encode'] = le.fit_transform(olympics_com
plete_subset['Team'])
olympics_complete_subset['Event_encode'] = le.fit_transform(olympics_co
mplete_subset['Event'])


#storing Team names and corresponding  encoded numerical values into ne
w csv file after sorting them according to Team name
TeamKeys=olympics_complete_subset[['Team', 'Team_encode']].copy()
TeamKeys.drop_duplicates(subset ="Team", inplace = True)
TeamKeys.sort_values("Team", axis = 0, ascending = True, inplace = True
, na_position ='last')
TeamKeys.to_csv('/content/drive/My Drive/DataMiningProjecct/Preprocesse
d-datasets/keysToTeam.csv')
print(TeamKeys.head())


#storing Event names and corresponding  encoded numerical values into n
ew csv file after sorting them according to Event name
EventKeys=olympics_complete_subset[['Event' , 'Event_encode']].copy()
EventKeys.drop_duplicates(subset ="Event", inplace = True)
EventKeys.sort_values("Event", axis = 0, ascending = True, inplace = Tr
ue, na_position ='last')
EventKeys.to_csv('/content/drive/My Drive/DataMiningProjecct/Preprocess
ed-datasets/keysToEvent.csv')
print(EventKeys.head())


#Creating a new dataframe to store keys to Sex and store it in new csv
file
SexKeys = pd.DataFrame( {'Sex': ['Male','Female'], 'Sex_encode': [1,0]
}, index =[1,2] )
SexKeys.to_csv('/content/drive/My Drive/DataMiningProjecct/Preprocessed
-datasets/keysToSex.csv')
print(SexKeys)


'''after mapping the key value pairs, dropping the team and Event colum
ns from final dataset and changing
```

```
names of Team_encode to Team and Event_encode to Event in the original
dataframe'''
olympics_complete_subset.drop(['Team'] , axis=1, inplace=True)
olympics_complete_subset.drop(['Event'] , axis=1, inplace=True)
olympics_complete_subset.rename(columns = {'Team_encode': 'Team'}, inpl
ace = True)
olympics_complete_subset.rename(columns = {'Event_encode': 'Event'}, in
place = True)


olympics_complete_subset.drop(['index'] , axis=1, inplace=True)
#Converting the final dataset into a new csv file
olympics_complete_subset.to_csv('/content/drive/My Drive/DataMiningProj
ecct/Preprocessed-datasets/olympics_final_final.csv');
print(olympics_complete_subset.head())
```

2) Clustering.ipynb

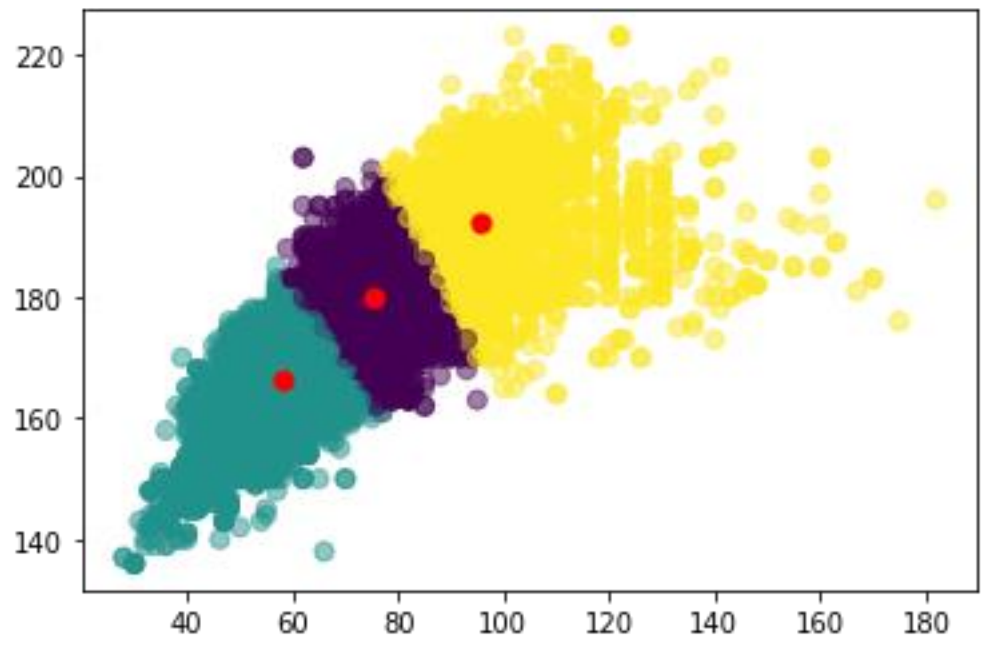
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans


data= pd.read_csv('/content/drive/My Drive/DataMiningProjecct/Preproces
sed-datasets/olympics_final_final.csv')


data= data[data['Medal_Won']!=0]
data1=pd.DataFrame(data[['Age','Weight','Height','Medal_Won']])


kmeans = KMeans(n_clusters=3).fit(data1)
centroid = kmeans.cluster_centers_
print(centroid)
```



```
[[ 25.6737747   75.12628562 180.12469685   1.98104407]
 [ 24.31509065  58.27857051 166.32173061   1.97382956]
 [ 26.18893823  95.70276136 192.23716265   2.02099314]]
```

```
plt.scatter(data1['Weight'],data1['Height'],c=kmeans.labels_.astype(flo
at),s=50,alpha=0.5)
```

```
plt.scatter(centroid[:,1],centroid[:,2], c='red',s=50)
plt.show()
```



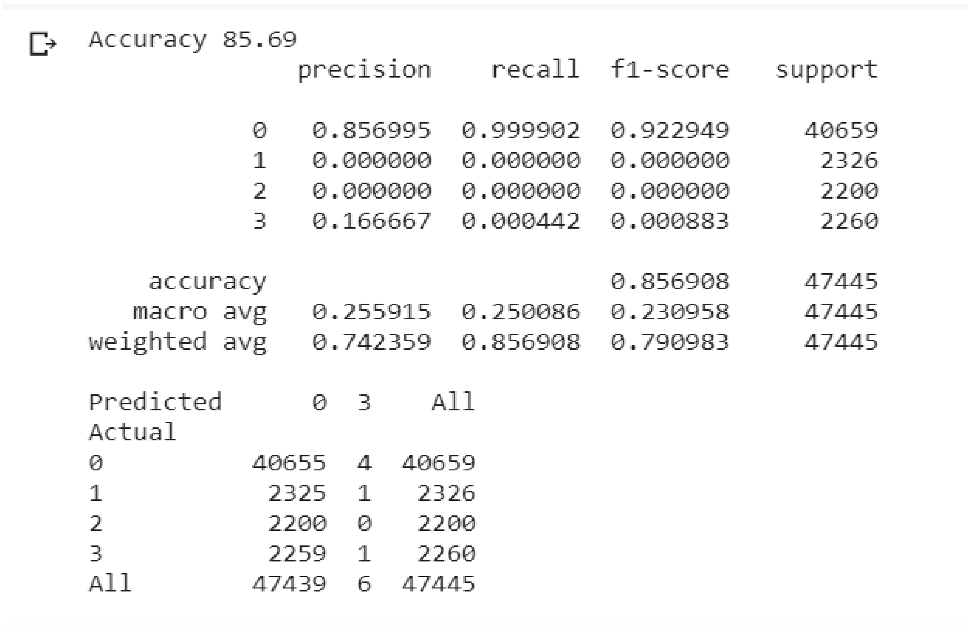
3) Classification.ipynb

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import TimeSeriesSplit, train_test_split
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import matplotlib
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.svm import LinearSVC
import pylab as pl
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings('ignore')
```

```
data= pd.read_csv('/content/drive/My Drive/DataMiningProjecct/Preproces
sed-datasets/olympics_final_final.csv')
data1= pd.DataFrame(data[['Sex','Age','Height','Weight','Team','Event']
])

X_train, X_test, Y_train, Y_test = train_test_split(data1, data['Medal_
Won'],test_size=0.30, random_state=99)
```

Algo:1- Logistic Regression

```
lr = LogisticRegression()

lr.fit(X_train, Y_train)
Y_pred = lr.predict(X_test)
sk_report = classification_report(digits=6, y_true=Y_test, y_pred=Y_pre
d)
print("Accuracy", round(accuracy_score(Y_pred, Y_test)*100,2))
#print(sk_report)
print(pd.crosstab(Y_test, Y_pred, rownames=['Actual'], colnames=['Predi
cted'], margins=True))
```

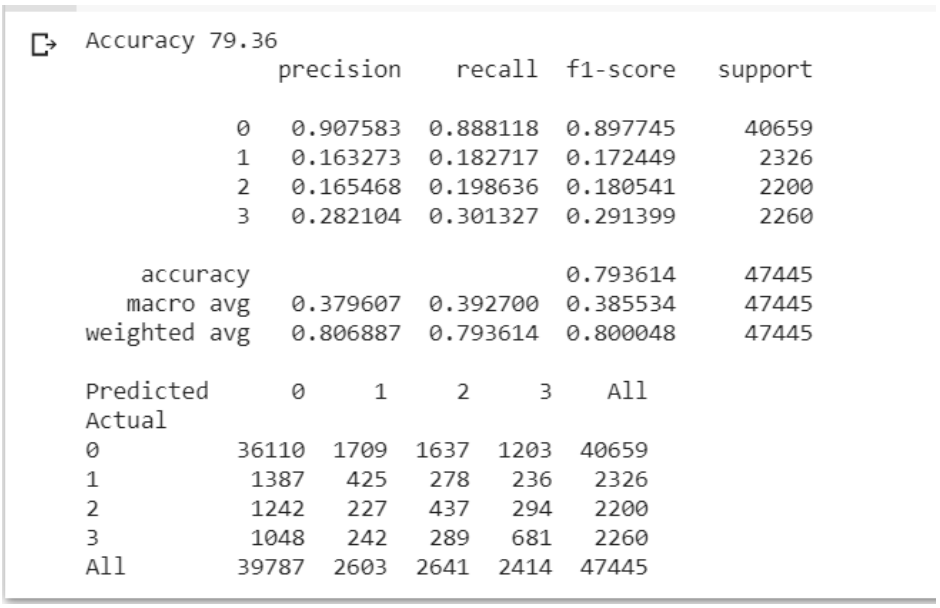


```
Accuracy 85.69
              precision    recall  f1-score   support

           0   0.856995  0.999902  0.922949     40659
           1   0.000000  0.000000  0.000000      2326
           2   0.000000  0.000000  0.000000      2200
           3   0.166667  0.000442  0.000883      2260

    accuracy                       0.856908     47445
   macro avg   0.255915  0.250086  0.230958     47445
weighted avg   0.742359  0.856908  0.790983     47445

Predicted      0  3    All
Actual
0          40655  4  40659
1           2325  1   2326
2           2200  0   2200
3           2259  1   2260
All        47439  6  47445
```

Algo:2- Decision Tree Algorithm

```
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree1 = round(decision_tree.score(X_test, Y_test) * 100, 2
)
```

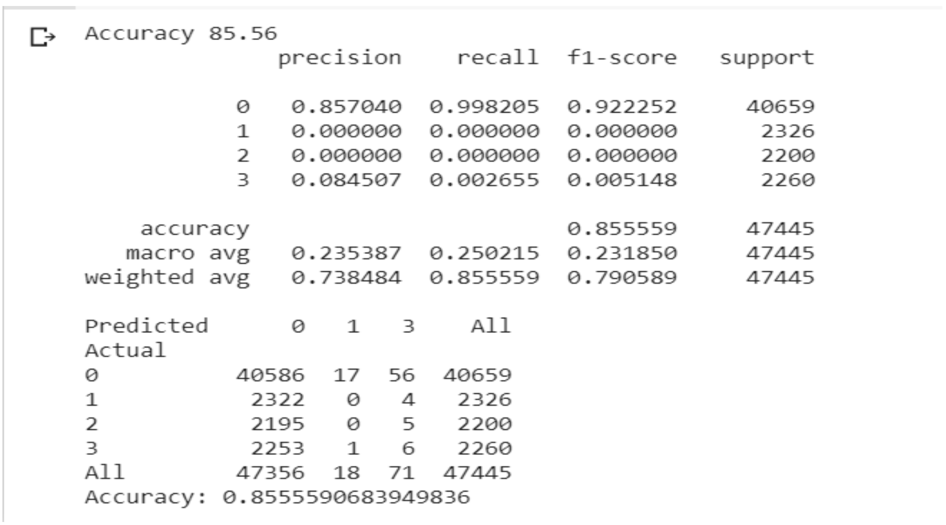
```
sk_report = classification_report(digits=6, y_true=Y_test, y_pred=Y_pre
d)
print("Accuracy", acc_decision_tree1)
print(sk_report)
### Confusion Matrix
print(pd.crosstab(Y_test, Y_pred, rownames=['Actual'], colnames=['Predi
cted'], margins=True))
```



```
Accuracy 79.36
              precision    recall  f1-score   support

           0   0.907583  0.888118  0.897745     40659
           1   0.163273  0.182717  0.172449      2326
           2   0.165468  0.198636  0.180541      2200
           3   0.282104  0.301327  0.291399      2260

    accuracy                       0.793614     47445
   macro avg   0.379607  0.392700  0.385534     47445
weighted avg   0.806887  0.793614  0.800048     47445

Predicted      0     1     2     3    All
Actual
0          36110  1709  1637  1203  40659
1           1387   425   278   236   2326
2           1242   227   437   294   2200
3           1048   242   289   681   2260
All        39787  2603  2641  2414  47445
```

Algorithm:3- Naïve Bayes Algorithm

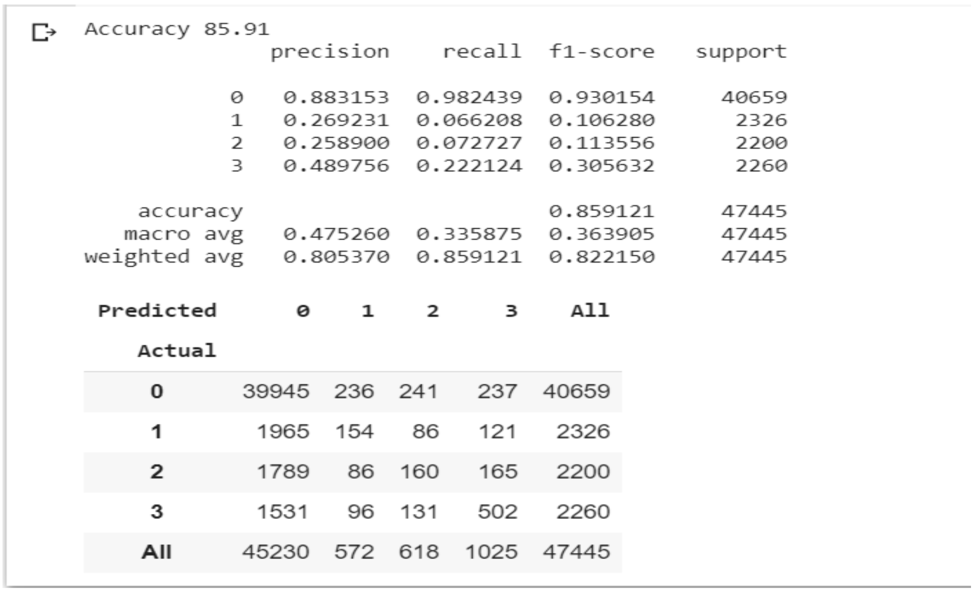
```
#Create a Gaussian Classifier
model = GaussianNB()
# Train the model using the training sets
model.fit(X_train,Y_train)
#Predict Output
Y_pred = model.predict(X_test)
sk_report = classification_report(digits=6, y_true=Y_test, y_pred=Y_pre
d)
print("Accuracy", round(accuracy_score(Y_pred, Y_test)*100,2))
print(sk_report)
print(pd.crosstab(Y_test, Y_pred, rownames=['Actual'], colnames=['Predi
cted'], margins=True))
print("Accuracy:",metrics.accuracy_score(Y_test, Y_pred))
```



```
Accuracy 85.56
              precision    recall  f1-score   support

           0   0.857040  0.998205  0.922252     40659
           1   0.000000  0.000000  0.000000      2326
           2   0.000000  0.000000  0.000000      2200
           3   0.084507  0.002655  0.005148      2260

    accuracy                       0.855559     47445
   macro avg   0.235387  0.250215  0.231850     47445
weighted avg   0.738484  0.855559  0.790589     47445

Predicted      0   1   3    All
Actual
0          40586  17  56  40659
1           2322   0   4   2326
2           2195   0   5   2200
3           2253   1   6   2260
All        47356  18  71  47445
Accuracy: 0.8555590683949836
```

Algorithm:4- Random Forest Algorithm

```
random_forest = RandomForestClassifier(n_estimators=200)
random_forest.fit(X_train,Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_test, Y_test)
acc_random_forest1 = round(random_forest.score(X_test, Y_test) * 100, 2
)
sk_report = classification_report(
    digits=6,
    y_true=Y_test,
    y_pred=Y_pred)
print("Accuracy" , acc_random_forest1)
print(sk_report)
pd.crosstab(Y_test, Y_pred, rownames=['Actual'], colnames=['Predicted']
, margins=True)
```



```
Accuracy 85.91
              precision    recall  f1-score   support

           0   0.883153  0.982439  0.930154     40659
           1   0.269231  0.066208  0.106280      2326
           2   0.258900  0.072727  0.113556      2200
           3   0.489756  0.222124  0.305632      2260

    accuracy                       0.859121     47445
   macro avg   0.475260  0.335875  0.363905     47445
weighted avg   0.805370  0.859121  0.822150     47445
```

Predicted	**0**	**1**	**2**	**3**	**All**
Actual					
0	39945	236	241	237	40659
1	1965	154	86	121	2326
2	1789	86	160	165	2200
3	1531	96	131	502	2260
All	45230	572	618	1025	47445

RESULTS AND DISCUSSIONS

- It is seen that the Logistic Regression algorithm gives a pretty good accuracy score, yet when we analyse the prediction results, it proves to be inefficient in our scenario.
- When we look at the results of Naïve Bayes algorithm, its accuracy score is quite good. But when we analyse its confusion matrix, this algorithm is also found to be less efficient.
- The accuracy score of Random Forest algorithm as well as its matrix gives quite good results. Yet since Random forest algorithm is quite complex and takes much time in generating the results, we may look for another possible alternative.
- Looking at the results of Decision tree algorithm, although its accuracy score is not as high as random forest and Naïve Bayes, yet it shows a very good confusion matrix and is also less complex than random forest.

Thus, we can conclude Decision Tree algorithm to be the most efficient algorithm in this case.

REFERENCES

[1] Models and Data Mining Algorithms for Solving Classification Problems

Author(s):Zayar Aung ; Ilya Sergeevich Mihailov ; Ye Thu Aung.

[2] A Review on Different Data Mining Algorithms and Selection Methods

Author(s): Kavita Kaithwas, Prashant Borkar.

[3] Research on Intelligent Tutoring System Based on Data-Mining Algorithms

Author(s): Yixuan Chen, Yang Zhang.

[4] Performance Evaluation of Mahout Clustering Algorithms Using a Twitter Streaming Dataset.

Author(s): Fatos Xhafa, Adriana Bogza, Santi Caballé.

[5] Enhanced mining association rule algorithm with reduced time & space

Complexity.

Author(s): Punit Mundra, Amit K Maurya, Sanjay Singh.

[6] Improved algorithm based on Sequential pattern mining of big data set.
Author(s): Peng Huang.

[7] Predicting rank for scientific research papers using supervised learning.
Author(s); MohamedEl Mohadab, Belaid Bouikhalene, Said Safi.

[8] A hybrid data mining model of feature selection algorithms and ensemble learning classifiers for credit scoring.
Author(s): Fatemeh Nemati Koutanaeia, Hedieh Sajedib, Mohammad Khanbabaeic.

[9] Comparison of different machine learning methods on Wisconsin dataset.
Author(s): Ivancakova, Juliana, Babic, Frantisek, Butka, Peter.

[10] Concept drift in Streaming Data Classification: Algorithms, Platforms and Issues.
Author(s): Janardan, Shikha Mehta.