SmallSEOTools

PLAGIARISM SCAN REPORT

Words	446	Date	April 03,2021
Characters	6106	Excluded URL	

0%	100%	0	1
Plagiarism	Unique	Plagiarized Sentences	Unique Sentences

Content Checked For Plagiarism

```
import networkx as nx
import matplotlib.pyplot as plt
#nx.graph()
graph =nx.read_weighted_edgelist("reachability.txt", comments='#', delimiter=None, create_using=None , nodetype=int,
encoding='utf-8')
graph
for e in graph.edges():
graph[e[0]][e[1]]['weight'] = 1


print(nx.info(graph))
import random
selectedNodes = set()
G = graph.copy()
for i in list(G.nodes):
if i > 19:
G.remove_node(i)
print(nx.info(G))
#Visualising Orginal graph
nx.draw(G, with_labels=True, node_size=1500, node_color="green", pos=nx.circular_layout(G))
plt.show()
print("Average Clustering Coefficient of Graph : "+str(nx.average_clustering(graph)))
print("Transitivity of the Graph : "+str(nx.transitivity(graph)))
print("Average Shortest Path Length : "+str(nx.average_shortest_path_length(graph)))
#Constructing Erdos Renyi Graph
erdos= nx.erdos_renyi_graph(20,0.455)
nx.draw(erdos, with_labels=True, node_size=1500, node_color="skyblue", pos=nx.circular_layout(erdos))
plt.show()
print(nx.info(erdos))
#Analysing Erdos Renyi Graph
print("Average Clustering Coefficient of Graph : "+str(nx.average_clustering(erdos)))
print("Transitivity of the Graph : "+str(nx.transitivity(erdos)))
print("Average Shortest Path Length : "+str(nx.average_shortest_path_length(erdos)))
#Constructing Watts Strogatz Graph
ws = nx.connected_watts_strogatz_graph(n = 20,k=9,p = 0.01)
nx.draw(ws, with_labels=True, node_size=1500, node_color="orange", pos=nx.circular_layout(ws))
plt.show()
print(nx.info(ws))
#Analysing Watts Strogatz Graph
```

```
print("Average Clustering Coefficient of Graph : "+str(nx.average_clustering(ws)))
print("Transitivity of the Graph : "+str(nx.transitivity(ws)))
print("Average Shortest Path Length : "+str(nx.average_shortest_path_length(ws)))
#Constructing Barabasi Albert Graph
ba= nx.barabasi_albert_graph(20,6)
nx.draw(ba, with_labels=True, node_size=1500, node_color="pink", pos=nx.circular_layout(ba))
plt.show()
print(nx.info(ba))
#Analysing Barabasi Albert Graph
print("Average Clustering Coefficient of Graph : "+str(nx.average_clustering(ba)))
print("Transitivity of the Graph : "+str(nx.transitivity(ba)))
print("Average Shortest Path Length : "+str(nx.average_shortest_path_length(ba)))
nx.draw(pc, with_labels=True, node_size=1500, node_color="yellow", pos=nx.circular_layout(pc))
plt.show()
print(nx.info(pc))
#Analysing Power Law Cluster Graph
print("Average Clustering Coefficient of Graph : "+str(nx.average_clustering(pc)))
print("Transitivity of the Graph : "+str(nx.transitivity(pc)))
print("Average Shortest Path Length : "+str(nx.average_shortest_path_length(pc)))
"""##Section 3"""
dataset = pd.read_csv("airlines.csv")
dataset.head()
#total rows and colums in the dataset
dataset1=dataset[dataset['Time.Year']==2003]
#[203 rows x 24 columns]
dataset1.shape
dataset1.info()
X= dataset1.iloc[:, [6,10]].values #Statistics.# of Delays.Carrier &&Statistics.# of Delays.Weather
#print(X)// if needed
#uses Elbow Mmethod
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
kmeans = KMeans(n_clusters= i, init='k-means++', random_state=0)
kmeans.fit(X)
wcss.append(kmeans.inertia_)
# model creation
kmeansmodel = KMeans(n_clusters= 5, init='k-means++', random_state=0)
y_kmeans= kmeansmodel.fit_predict(X)
#Visualizing all the clusters
plt.xlabel('Statistics.# of Delays.Carrier')
plt.ylabel('Statistics.# of Delays.Weather')
plt.legend()
plt.show()
dataset2=dataset[dataset['Time.Year']==2005]
dataset2.shape
X= dataset2.iloc[:, [13,16]].values#
#print(X)
#KMeans Algorithm decides the optimum cluster number
#uses Elbow Mmethod
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
kmeans = KMeans(n_clusters= i, init='k-means++', random_state=0)
kmeans.fit(X)
wcss.append(kmeans.inertia_)
#model creation
kmeansmodel = KMeans(n_clusters= 5, init='k-means++', random_state=0)
y_kmeans= kmeansmodel.fit_predict(X)
plt.title('Clusters Year:2005')
plt.xlabel('Statistics.# of Flights Cancelled')
plt.ylabel('Statistics.# of Flights On-time')
plt.legend()
plt.show()
dataset3=dataset[dataset['Time.Year']==2007]
```

```
dataset3.shape
X= dataset3.iloc[:, [14,16]].values
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
kmeans = KMeans(n_clusters= i, init='k-means++', random_state=0)
kmeans.fit(X)
wcss.append(kmeans.inertia_)
#model creation
kmeansmodel = KMeans(n_clusters= 5, init='k-means++', random_state=0)
y_kmeans= kmeansmodel.fit_predict(X)
#Visualizing all the clusters
plt.title('Clusters Year:2007')
plt.xlabel('Statistics.# of Flights Delayed')
plt.ylabel('Statistics.# of Flights On-time')
plt.legend()
plt.show()
dataset4=dataset[dataset['Time.Year']==2009]
dataset4.shape
X= dataset4.iloc[:, [16,17]].values
#uses Elbow Mmethod
from sklearn.cluster import KMeans
wcss=[]
for i in range(1,11):
kmeans = KMeans(n_clusters= i, init='k-means++', random_state=0)
kmeans.fit(X)
wcss.append(kmeans.inertia_)
#model creation
kmeansmodel = KMeans(n_clusters= 5, init='k-means++', random_state=0)
y_kmeans= kmeansmodel.fit_predict(X)
#Visualizing all the clusters
plt.title('Clusters Year:2009')
plt.xlabel('Statistics.# of Flights ON-time')
plt.ylabel('Statistics.# of Flights TOTAL')
plt.legend()
plt.show()
```

Sources	Similarity