sena_project.py

Scanned on: 18:22 November 10, 2022 UTC

Overall Similarity Score

Results Found

Total Words in Text

Identical Words	20
Words with Minor Changes	1
Paraphrased Words	0
Omitted Words	0

sena_project.py

Scanned on: 18:22 November 10, 2022 UTC

Results

Sources that matched your submitted document.

Assignment1 *https://deepnote.com/@samuelivarsson-486d/Assignment1-a1af7cc6-34c...*	3%
Creating subplots with a for loop Pandas Python - Stack Ove... *https://stackoverflow.com/questions/72382074/creating-subplots-with-a-f...*	2%
Network analysis in Python *https://autogis-site.readthedocs.io/en/2019/notebooks/L6/network-analysi...* network-analysis.html	2%
How to plot a scatter with Pandas and Matplotlib? | EasyTw... *https://www.easytweaks.com/create-python-scatter-plot/*	2%
Solved import matplotlib.pyplot as plt import pandas as | C... *https://www.chegg.com/homework-help/questions-and-answers/import-m...*	2%

IDENTICAL

Identical matches are one to one exact wording in the text.

MINOR CHANGES

Nearly identical with different form, ie "slow" becomes "slowly".

PARAPHRASED

Close meaning but different words used to convey the same message.

Unsure about your report?

The results have been found after comparing your submitted text to online sources, open databases and the Copyleaks internal database. For any questions about the report contact us on support@copyleaks.com

Learn more about different kinds of plagiarism here

sena_project.py

Scanned on: 18:22 November 10, 2022 UTC

Scanned Text

Your text is highlighted according to the matched content in the results above.

IDENTICAL MINOR CHANGES PARAPHRASED

```
import networkx as nx
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

df = pd.read_csv('Edges.csv')
df.head()
df.columns
G = nx.from_pandas_edgelist(df,source='Source',target='Target')
plt.figure(figsize=(20,20))
nx.draw(G)
plt.hist([v for k,v in nx.degree(G)])
nx.cluster.average_clustering(G)

try:
nx.diameter(G)
except Exception as e:
print('Infinite Diameter')
len(G)

def graph_to_edge_matrix(G):
edge_mat = np.zeros((len(G),len(G)),dtype=int)
for node in G:
for neighbor in G.neighbors(node):
edge_mat[node-1][neighbor-1]=1
edge_mat[node-1][node-1]=1
return edge_mat

edge_mat = graph_to_edge_matrix(G)
edge_mat

from sklearn import cluster
from sklearn.metrics.cluster import normalized_mutual_info_score
from sklearn.metrics.cluster import adjusted_rand_score

k_clusters = 6
results = []
algorithms={}

algorithms['kmeans'] = cluster.KMeans(n_clusters=k_clusters,n_init=200)
```

```
algorithms['agglom'] = cluster.AgglomerativeClustering(n_clusters=k_clusters,linkage="ward")

model = algorithms['kmeans'].fit(edge_mat)

results = list(model.labels_)

print(set(results))

print(f"The number of people in cluster 0 is", results.count(0))
print(f"The number of people in cluster 1 is", results.count(1))
print(f"The number of people in cluster 2 is", results.count(2))
print(f"The number of people in cluster 3 is", results.count(3))
print(f"The number of people in cluster 4 is", results.count(4))
print(f"The number of people in cluster 5 is", results.count(5))

clusters = {

}

df['Relation'].value_counts()

clusters = {
'Sci&Tech and Mkt&Fin':[],
'Sci&Tech and Mkt&HR':[],
'Comm&Mgmt and Mkt&Fin':[],
'Comm&Mgmt and Mkt&HR':[],
'Others and Mkt&HR':[],
'Others and Mkt&Fin':[]
}

df = pd.read_csv('Placement_Data_Full_Class.csv')
df.head()

df.columns

df['degree_t'].value_counts()

for index,row in df.iterrows():
if row['degree_t'] == 'Sci&Tech' and row['specialisation']=='Mkt&Fin':
clusters['Sci&Tech and Mkt&Fin'].append(row['sl_no'])
elif row['degree_t'] == 'Sci&Tech' and row['specialisation']=='Mkt&HR':
clusters['Sci&Tech and Mkt&HR'].append(row['sl_no'])
elif row['degree_t'] == 'Comm&Mgmt' and row['specialisation']=='Mkt&Fin':
clusters['Comm&Mgmt and Mkt&Fin'].append(row['sl_no'])
elif row['degree_t'] == 'Comm&Mgmt' and row['specialisation']=='Mkt&HR':
clusters['Comm&Mgmt and Mkt&HR'].append(row['sl_no'])
elif row['degree_t'] == 'Others' and row['specialisation']=='Mkt&HR':
clusters['Others and Mkt&HR'].append(row['sl_no'])
else:
clusters['Others and Mkt&Fin'].append(row['sl_no'])

print(len(clusters['Sci&Tech and Mkt&Fin']))
print(len(clusters['Sci&Tech and Mkt&HR']))
print(len(clusters['Comm&Mgmt and Mkt&Fin']))
print(len(clusters['Comm&Mgmt and Mkt&HR']))
print(len(clusters['Others and Mkt&HR']))
print(len(clusters['Others and Mkt&Fin']))

clusters['Sci&Tech and Mkt&Fin']

dataset = {
'Sl_no':[],
'Branch And Specialization':[],
'Placement Status':[],
'Salary':[]
}
```

```
for index,row in df.iterrows():
dataset['Sl_no'].append(row['sl_no'])
dataset['Branch And Specialization'].append(row['degree_t']+' '+row['specialisation'])
dataset['Placement Status'].append(row['status'])
dataset['Salary'].append(row['salary'])

len(df)

len(dataset['Salary'])

Dataframe = pd.DataFrame(dataset)
Dataframe.head()

Dataframe.isnull().sum()

Dataframe.fillna(value=0.0,inplace=True)
Dataframe.head()

Dataframe['Branch And Specialization'].value_counts()

Dataframe[Dataframe['Branch And Specialization']=='Sci&Tech Mkt&HR']['Placement
Status'].value_counts()['Placed']

lst={
'Sci&Tech Mkt&HR':Dataframe[Dataframe['Branch And Specialization']=='Sci&Tech Mkt&HR']['Placement
Status'].value_counts()['Placed'],
'Sci&Tech Mkt&Fin':Dataframe[Dataframe['Branch And Specialization']=='Sci&Tech Mkt&Fin']['Placement
Status'].value_counts()['Placed'],
'Comm&Mgmt Mkt&Fin':Dataframe[Dataframe['Branch And Specialization']=='Comm&Mgmt Mkt&Fin']
['Placement Status'].value_counts()['Placed'],
'Comm&Mgmt Mkt&HR':Dataframe[Dataframe['Branch And Specialization']=='Comm&Mgmt Mkt&HR']
['Placement Status'].value_counts()['Placed'],
'Others Mkt&HR':Dataframe[Dataframe['Branch And Specialization']=='Others Mkt&HR']['Placement
Status'].value_counts()['Placed'],
'Others Mkt&Fin':Dataframe[Dataframe['Branch And Specialization']=='Others Mkt&Fin']['Placement
Status'].value_counts()['Placed']
}
lst

avg_salary = []

# Average Salary of Sci&Tech Mkt&HR
avg_salary.append((Dataframe[Dataframe['Branch And Specialization']=='Sci&Tech Mkt&HR']
['Salary'].sum()/len(Dataframe[Dataframe['Branch And Specialization']=='Sci&Tech Mkt&HR'])).round(2))

# Average Salary of Comm&Mgmt Mkt&Fin
avg_salary.append((Dataframe[Dataframe['Branch And Specialization']=='Comm&Mgmt Mkt&Fin']
['Salary'].sum()/len(Dataframe[Dataframe['Branch And Specialization']=='Comm&Mgmt
Mkt&Fin'])).round(2))

# Average Salary of Comm&Mgmt Mkt&HR
avg_salary.append((Dataframe[Dataframe['Branch And Specialization']=='Comm&Mgmt Mkt&HR']
['Salary'].sum()/len(Dataframe[Dataframe['Branch And Specialization']=='Comm&Mgmt
Mkt&HR'])).round(2))

# Average Salary of Sci&Tech Mkt&Fin
avg_salary.append((Dataframe[Dataframe['Branch And Specialization']=='Sci&Tech Mkt&Fin']
['Salary'].sum()/len(Dataframe[Dataframe['Branch And Specialization']=='Sci&Tech Mkt&Fin'])).round(2))

# Average Salary of Others Mkt&HR
avg_salary.append((Dataframe[Dataframe['Branch And Specialization']=='Others Mkt&HR']
['Salary'].sum()/len(Dataframe[Dataframe['Branch And Specialization']=='Others Mkt&HR'])).round(2))

# Average Salary of Others Mkt&Fin
avg_salary.append((Dataframe[Dataframe['Branch And Specialization']=='Others Mkt&Fin']
['Salary'].sum()/len(Dataframe[Dataframe['Branch And Specialization']=='Others Mkt&Fin'])).round(2))
```

```
avg_salary

bxaxis = ['Sci&Tech Mkt&HR','Sci&Tech Mkt&Fin','Comm&Mgmt Mkt&Fin','Comm&Mgmt Mkt&HR','Others Mkt&HR','Others Mkt&Fin']
byaxis = [lst[_] for _ in lst]

plt.figure(figsize=(14,5))
plt.bar(bxaxis, byaxis, color='g')
plt.title("Number of Placed Students")
plt.xlabel("Department")
plt.ylabel("Number of Students")
plt.show()

plt.figure(figsize=(14,5))
gxaxis = ['Sci&Tech Mkt&HR','Comm&Mgmt Mkt&Fin','Comm&Mgmt Mkt&HR','Sci&Tech Mkt&Fin','Others Mkt&HR','Others Mkt&Fin']
plt.bar(gxaxis, avg_salary, color='b')
plt.title("Student's Average Salary")
plt.xlabel("Department")
plt.ylabel("Average Salary")
plt.show()

print("Hence from this analysis we can know that " + gxaxis[avg_salary.index(max(avg_salary))] + " has high Average Salary")
print("Hence from this analysis we can know that " + bxaxis[byaxis.index(max(byaxis))] + " has Highest Placement")
```