New Scan 5:27 AM

Scanned on: 5:27 November 11, 2022 UTC

12.8%	4	694
Overall Similarity Score	Results Found	Total Words in Text

Identical Words	84
Words with Minor Changes	5
Paraphrased Words	0
Omitted Words	0

New Scan 5:27 AM

Scanned on: 5:27 November 11, 2022 UTC

Results

Sources that matched your submitted document.

Copyleaks Internal Database	10%
Copyleaks Internal Database	9%
Copyleaks Internal Database	2%
Copyleaks Internal Database	1%

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

COPYLEAKS

New Scan 5:27 AM

Scanned on: 5:27 November 11, 2022 UTC

Scanned Text

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

● IDENTICAL ● MINOR CHANGES ● PARAPHRASED

```
import pandas as pd
import numpy as np
import json
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import seaborn as sns

from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import mean_squared_error



nodes = json.load(open('node_list.json', 'rb'))
edges = json.load(open('edge_list.json', 'rb'))


df= nx.DiGraph()


[df.add_node(n['id'], nation=n['nation']) for n in nodes];


[df.add_edge(edge['from'], edge['to'], type=edge['type']) for edge in edges];


colour= {
'Mondstadt': '#389695',
'Liyue': '#d1a158',
'Inazuma': '#816ab5',
'Snezhnaya': '#7ad7f0'
}


node_colors = []
for node, nation in list(df.nodes(data="nation")):
if nation in colour:
node_colors.append(colour[nation])
```

```
else:
node_colors.append('#979392')


%%time
np.random.seed(2021)
pos = nx.spring_layout(df, k=0.3, iterations=25)


plt.figure(1,figsize=(15,15))
nx.draw(df,
pos = pos,
node_size=2000,
node_color=node_colors,
with_labels=True,
font_size = 11,
font_color='black')
legend_tiles = [mpatches.Patch(color="#389695", label="Mondstadt"),
mpatches.Patch(color="#d1a158", label="Liyue"),
mpatches.Patch(color="#816ab5", label="Inazuma"),
mpatches.Patch(color="#7ad7f0", label="Snezhnaya"),
mpatches.Patch(color="#979392", label="Traveler"),
]


plt.title("Genshin Impact Character Social Network")
plt.legend(handles=legend_tiles, loc="upper left")
plt.show()


df.number_of_nodes()


df.number_of_edges()


dir_nodes_df = pd.DataFrame(data=nodes, columns=['id', 'nation'])
dir_nodes_df['in_degree'] = dir_nodes_df['id'].apply(lambda n: df.in_degree(n))
dir_nodes_df.sort_values('in_degree', ascending=False).head(10)


dir_nodes_df['in_degree'].mean()


dir_nodes_df['out_degree'] = dir_nodes_df['id'].apply(lambda n: df.out_degree(n))
dir_nodes_df.sort_values('out_degree', ascending=False).head(10)


dir_nodes_df['out_degree'].mean()


pr = nx.pagerank(df)


dir_nodes_df['page_rank'] = dir_nodes_df['id'].apply(lambda n: pr[n])
dir_nodes_df[['id', 'nation', 'page_rank']].sort_values('page_rank', ascending=False).head(10)


undir = df.to_undirected(reciprocal=True)


plt.figure(1,figsize=(15,15))
nx.draw(undir,
nodelist=undir.nodes,
pos = pos,
node_size=2000,
with_labels=True,
font_size = 11,
font_color='black')
plt.show()
isolates = list(nx.isolates(undir))
undir.remove_nodes_from(isolates)


undir.number_of_nodes()


undir.number_of_edges()


undir_df = dir_nodes_df[['id', 'nation']].copy()
undir_df = undir_df[~undir_df['id'].isin(isolates)]


undir_df['degree'] = undir_df['id'].apply(lambda n: len(undir.edges(n)))
undir_df.sort_values('degree', ascending=False).head(10)


close_cen = nx.closeness_centrality(undir)
undir_df['closeness'] = undir_df['id'].apply(lambda n: close_cen[n])
undir_df.sort_values('closeness', ascending=False).head(10)


bet_cen = nx.betweenness_centrality(undir)
undir_df['betweenness'] = undir_df['id'].apply(lambda n: bet_cen[n])
undir_df.sort_values('betweenness', ascending=False).head(10)


eigen_cen = nx.eigenvector_centrality(undir)
undir_df['eigen'] = undir_df['id'].apply(lambda n: eigen_cen[n])
undir_df[['id','nation','degree','eigen']].sort_values('eigen', ascending=False).head(10)


ranked_df = undir_df[['id', 'nation']].copy()
undir_df = undir_df.sort_values('degree', ascending=False)
ranked_df['degree_rank'] = undir_df['degree'].rank(method='first', ascending=False).astype(int)


undir_df = undir_df.sort_values('closeness', ascending=False)
ranked_df['closeness_rank'] = undir_df['closeness'].rank(method='first', ascending=False).astype(int)


undir_df = undir_df.sort_values('betweenness', ascending=False)
ranked_df['betweenness_rank'] = undir_df['betweenness'].rank(method='first',
ascending=False).astype(int)


undir_df = undir_df.sort_values('eigen', ascending=False)
ranked_df['eigen_rank'] = undir_df['eigen'].rank(method='first', ascending=False).astype(int)


ranked_df['average_rank'] = ranked_df.mean(numeric_only=True, axis=1)
ranked_df.sort_values('average_rank')


ranked_df


hub_scores, auth_scores = nx.hits(df)
hub_centrality_df = pd.DataFrame(hub_scores.items(),
columns=["node", "hub_centrality"])
auth_centrality_df = pd.DataFrame(auth_scores.items(),
columns=["node", "auth_centrality"])
hub_centrality_df.sort_values("hub_centrality", ascending=False).head(10)


auth_centrality_df.sort_values("auth_centrality", ascending=False).head(10)




path= '/content/drive/MyDrive/dataset/stronger.csv'


element ={"Pyro": "#750550", "Hydro": "#395B64", "Geo": "#4FA095", "Anemo": "#153462", "Electro" :
"#ABCE30", "Cryo": "#E97777"}
nation ={"Mondstadt": "#395B64", "Liyue": "#153462", "Snezhnaya": "#E97777"}
weapon ={"Sword": "#4FA095", "Bow": "#750550", "Catalyst": "#ABCE30", "Claymore":
"#E97777","Polearm" : "#153462"}
Type ={"Male": "#ABCE30", "Female": "#395B64", "Player's Choice": "#750550"}


final = dict(weapon)
final.update(element)
final.update(nation)
final.update(Type)


categorical = ['Nation', 'Sex', 'Weapon', 'Element']
numerical = ['ATK', 'DEF']


fig, ax = plt.subplots(2, 2, figsize=(20, 10))
for variable, subplot in zip(categorical, ax.flatten()):
sns.countplot(df[variable], ax=subplot, palette = final, order = df[variable].value_counts().index)
for label in subplot.get_xticklabels():
label.set_rotation(0)


snezhnaya = df["Nation"]!="Snezhnaya"
traveler = df["Name"]!="Traveler"


sns.relplot(
data=df[snezhnaya & traveler], x="DEF", y="ATK",
col="Nation", hue="Element", style="Element",
kind="scatter", s = 200, palette = final
)


sns.relplot(
data=df[snezhnaya & traveler], x="DEF", y="ATK",
col="Nation", hue="Weapon", style="Weapon",
kind="scatter", s = 200, palette = final
)


plt.figure(figsize=(20,20))
sns.scatterplot(x=df['DEF'], y=df['ATK'], hue = df["Element"], style = df["Weapon"], palette = final, s = 200);


for i in range(df.shape[0]):
plt.text(x=df.DEF[i]+2.5,y=(df.ATK[i]+5 if df.Name[i]=="Beidou" else df.ATK[i]-0.2),s=df.Name[i],
fontdict=dict(color='black',size=10),
bbox=dict(facecolor='black',alpha=0.1))
```

```
path= '/content/drive/MyDrive/dataset/genshin.csv'
df= pd.read_csv(path)
df.head()


columns= ['character_id', 'playable', 'vision', 'region', 'weapon_type',
'hp_90_90', 'atk_90_90', 'def_90_90', 'hp_80_90', 'atk_80_90',
'def_80_90', 'hp_80_80', 'atk_80_80', 'def_80_80', 'hp_70_80',
'atk_70_80', 'def_70_80', 'hp_70_70', 'atk_70_70', 'def_70_70',
'hp_60_70', 'atk_60_70', 'def_60_70', 'hp_60_60', 'atk_60_60',
'def_60_60', 'hp_50_60', 'atk_50_60', 'def_50_60', 'hp_50_50',
'atk_50_50', 'def_50_50', 'hp_40_50', 'atk_40_50', 'def_40_50',
'hp_40_40', 'atk_40_40', 'def_40_40', 'hp_20_40', 'atk_20_40',
'def_20_40', 'hp_20_20', 'atk_20_20', 'def_20_20', 'hp_1_20',
'atk_1_20', 'def_1_20']


data_x = df[columns].copy()
data_y = df['rarity'].copy()


data= PolynomialFeatures(2,include_bias=False)
data_x= data.fit_transform(data_x)


xtrain, xtest, ytrain, ytest = train_test_split( data_x, data_y, test_size=0.2, shuffle=True, random_state=42)
display((xtrain.shape, ytrain.shape))


x = df[columns].copy()
x = data.fit_transform(x)
y = np.array(df['rarity'].copy()).flatten()
regression = linear_model.LinearRegression()
regression.fit(xtrain, ytrain)
pred = regression.predict(x)
predicted = np.round(regression.predict(x)).astype(int)


print('Mean squared error: %.2f' % mean_squared_error(y, predicted))


def get_rarity(df,columns):
char= df.sample(1)
charx = char[columns].copy()
charx= data.fit_transform(charx)
chary = np.array(char['rarity'].copy())
return charx,chary


charx,chary = get_rarity(df, columns)
estimated = regression.predict(charx)
print('Actual value=',int(chary), 'Predicted value=', int(np.round(estimated)))


plt.figure(figsize=(5,5))
sb.scatterplot(predicted, y, alpha=0.15)
plt.plot(y, y, color='g', alpha=0.5)
plt.xlabel(' Overall Prediction')
plt.ylabel(' Overall True');
```