sena.py

Scanned on: 14:26 November 10, 2022 UTC

Overall Similarity Score

Results Found

Total Words in Text

Identical Words	45
Words with Minor Changes	1
Paraphrased Words	0
Omitted Words	0

sena.py

Scanned on: 14:26 November 10, 2022 UTC

Results

Sources that matched your submitted document.

Pavliuk_bakalavr.pdf
https://ela.kpi.ua/bitstream/123456789/45227/1/Pavliuk_bakalavr.pdf
Pavliuk_bakalavr.pdf
7%

Chandan 's Blog – Implementing and visualizing word2Vec u...
https://chandan-u.github.io/implementing-and-visualizing-word2vec-using...
implementing-and-visualizing-word2vec-using-tensorflow.html
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.py

Scanned on: 14:26 November 10, 2022 UTC

Scanned Text

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

IDENTICAL MINOR CHANGES PARAPHRASED

```
# -*- coding: utf-8 -*-
"""sena.ipynb

Automatically generated by Colaboratory.

Original file is located at
https://colab.research.google.com/drive/14AectWcp0qxRTw2tZGfjwESj9D7cD2HL
"""

import numpy as np
import pandas as pd
import plotly.graph_objs as go
from sklearn.model_selection import train_test_split
from sklearn import tree
from IPython.display import Image
from six import StringIO
import matplotlib.pyplot as plt
import pydotplus
from sklearn.ensemble import RandomForestClassifier
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

from sklearn.metrics import classification_report,confusion_matrix

from google.colab import drive
drive.mount('/content/drive')

input_file = "/content/drive/MyDrive/sena/globalterrorismdb_0522dist.csv"
df = pd.read_csv(input_file, header = 0,usecols=['iyear', 'imonth', 'iday', 'extended', 'country', 'country_txt',
'region', 'latitude', 'longitude','success', 'suicide','attacktype1',
'attacktype1_txt', 'targtype1', 'targtype1_txt', 'natlty1','natlty1_txt',
'weaptype1', 'weaptype1_txt' ,'nkill','multiple', 'individual',
'claimed','nkill','nkillter', 'nwound', 'nwoundte'])
df.head(5)
df.info()

# A dataframe with region South Asia (Afghanistan, Bangladesh, Bhutan, India, Maldives, Mauritius, Nepal,
Pakistan, Sri Lanka)
df_SouthAsia= df[df.region == 6]
df_SouthAsia.info()
```

```
df_SA = df_SouthAsia.drop([ 'region', 'claimed', 'nkillter', 'nwound','nwoundte'], axis=1)
df_SA.head()

df_SA.describe()

df_SA.plot(kind= 'scatter', x='longitude', y='latitude', alpha=0.4, figsize=(16,7))
plt.show()

df_SA['nkill'].fillna(0.686445, inplace=True)
df_SA['latitude'].fillna(47.004651, inplace=True)
df_SA['longitude'].fillna(10.921231, inplace=True)
df_SA['natlty1'].fillna(167.954530, inplace=True)

df_SA.info()

features = [ 'imonth', 'iday', 'extended', 'latitude', 'longitude', 'multiple','suicide','attacktype1','targtype1',
'individual', 'weaptype1', 'nkill']
X = df_SA.drop(['iyear', 'success','country', 'country_txt', 'attacktype1_txt','targtype1_txt','natlty1',
'natlty1_txt', 'weaptype1_txt'], axis=1)
y = df_SA['success']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

"""Decision tree - max depth(3)"""

y = df_SA['success'] #this is what we're trying to predict!
X = df_SA[features]
dtc = tree.DecisionTreeClassifier(max_depth=3)
dtc = dtc.fit(X_train,y_train)

#code to print the underlying configuration
dot_data = StringIO()
tree.export_graphviz(dtc, out_file=dot_data,feature_names=features)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())

dtc_pred = dtc.predict(X_test)
print(classification_report(y_test,dtc_pred))

print(confusion_matrix(y_test,dtc_pred))

"""Random Forest classification"""

rfclass = RandomForestClassifier(n_estimators=400)
rfclass = rfclass.fit(X_train, y_train)
rfclass_pred = rfclass.predict(X_test)

print(classification_report(y_test,rfclass_pred))

print(confusion_matrix(y_test,rfclass_pred))

#listing iportance of features
for name, score in zip(X_train[features], rfclass.feature_importances_):
print(name, score)

data = go.Bar(
y=['suicide', 'individual', 'extended', 'multiple', 'weaptype1', 'nkill', 'attacktype1','targtype1',
'imonth', 'iday', 'latitude', 'longitude'],
x=
[0.001182,0.002047,0.002392,0.011422,0.041777,0.085776,0.107362,0.109867,0.113829,0.167432,0.17358
3,0.18333],
orientation = 'h',
marker = dict(color = 'rgba(255,0,0, 0.6)', line = dict(width = 0.5)))

data = [data]
layout = go.Layout(title = 'Relative Importance of the Features in the Random Forest',
barmode='group', bargap=0.1, width=800,height=500,)
```

```
fig = go.Figure(data=data, layout=layout)
iplot(fig)

"""EXAMPLE OUTPUT FOR RANDOM FOREST CLASSIFICATION

"""

succeed_or_fail = RandomForestClassifier(n_estimators=400)
succeed_or_fail = rfclass.fit(X, y) #clf

month = 12 # in which month would the attack take place
day = 23 # on which day of the month would the attack take place
extended = 0 # 1=yes, 0=no
latitude = 48.8566
longitude = 2.3522
multiple = 0 # attack is part of a multiple incident (1), or not (0)
suicide = 0 # suicide attack (1) or not (0)
attackType = 3 # 9 categories
targetType = 7 # 22 categories
individual = 0 # known group/organization (1) or not (0)
weaponType = 6 # 13 categories
nkill = 0 # number of total casualties from the attack

outcome = (succeed_or_fail.predict([[(month),(day),(extended),(latitude),(longitude),(multiple),(suicide),
(attackType),(targetType),(individual),(weaponType),(nkill)]]))
if outcome == 1:
print(outcome)
print("The attack based on these features would be succesful.")
elif outcome == 0:
print(outcome)
print("The attack based on these features would NOT be succesful.")
```